НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_534.2.231____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018р.

Магістерська дисертація

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем
на тему: « Розробка програмної реалізації людино-машинного інтерфейсу HMI для візуалізації KNX-мережі, контролю та моніторингу її хостів»

Виконав: студент _6__курсу, групи_ТВ-71мп__
___Чернюк Андрій Миколайович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _доцент, к.е.н. Сегеда І. В._____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Програмне забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)

«____» _____ 2018р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Чернюку Андрію Миколайовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації: Розробка програмної реалізації людино-машинного інтерфейсу HMI для візуалізації KNX-мережі, контролю та моніторингу її хостів

Науковий керівник Сегеда Ірина Василівна к.е.н, доцент
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 15 березня 2018 року №1151-с
2. Строк подання студентом дисертації _____
3.Об’єкт дослідження структура KNX-мережі
4. Предмет дослідження інструменти для візуалізації структури KNX-мережі, контролю та моніторингу її хостів
5. Перелік питань, які потрібно розробити
1) проаналізувати структуру KNX-мережі та засоби її проектування;
2) проаналізувати проблематику проектування інтерфейсу користувача та принципи проектування інтерфейсу користувача;
3) сформулювати та проаналізувати базову функціональність для візуалізації структури KNX-мережі, контролю та моніторингу її хостів;

4) розробити програмне забезпечення для розробки людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів.

6. Орієнтований перелік ілюстративного матеріалу _____

1) Функціональність програмного забезпечення;

2) Діаграма послідовності роботи компонентів системи;

3) Взаємодія компонентів системи;

4) Структура програмного забезпечення

5) Інтерфейс користувача.

7. Орієнтований перелік публікацій _____

Сегеда І. В., Чернюк А. М., "Розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів"

8. Дата видачі завдання «_29_»__вересня____ 2017 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	29.09.2017	
2	Збір інформації	09.10.2017 – 25.01.2018	
3	Аналіз вимог завдання, вибір методів і засобів розв'язання поставленої задачі	26.01.2018 – 05.09.2018	
4	Підготовка публікацій	19.07.2018, 02.09.2018	
5	Підготовка доповідей на конференції	15.08.2018, 20.10.2018	
6	Підготовка дисертації	03.07.2018 – 07.12.2018	
7	Розробка програмного продукту	10.01.2018 – 09.10.2018	
8	Захист програмного продукту	24.10.2018	
9	Передзахист	28.11.2018	
10	Захист	18.12.2018	

Студент _____ Чернюк А. М.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Сегеда І. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Магістерська дисертація складається зі вступу, 6 основних розділів, висновку, переліку посилань з 25 найменувань, 2 додатків, 20 рисунків та 22 таблиць. Повний обсяг магістерської дисертації складає 88 сторінок, з яких перелік посилань займає 3 сторіноки, а додатки – 6 сторінок.

В сучасному світі кожна людина очікує більше зручності та комфорту, як у себе вдома так і на роботі. Сьогодні KNX є одним з найбільш поширених рішень для використання в середніх і великих системах автоматизації будинків, офісів і комерційних приміщень. Вимоги до зручності, безпеки та гнучкості сучасних будівель постійно зростають. При цьому, враховуючи постійний технологічний розвиток та підвищення автоматизації систем забезпечення комфорту, найбільш важливим питанням стає економія електроенергії. Саме тому попит на інтелектуальні системи, що відповідають цим умовам, постійно зростає.

Використання технології KNX може надати рішення, які при використанні звичайних методів створення подібних систем можуть бути реалізовані лише насилу. Контроль усіх застосувань в квартирі або будівлі може здійснюватися з однієї сенсорної панелі. Починаючи з систем опалювання, вентиляції і контролю доступу, і закінчуючи дистанційним керуванням усіма побутовими електроприладами - KNX відкриває абсолютно нові шляхи для підвищення комфорту, безпеки і економного енергоспоживання в квартирах і будівлях.

Метою дипломної роботи є розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів.

Призначення програмного візуалізації структури різних KNX-мереж, управління її елементами та моніторингу даних для подальношого аналізу та визначення більш оптимальних шляхів використання мережі.. Програмний продукт повинен забезпечити наступні можливості:

— створення проектів для використання;

— редагування проектів у випадку коли відбулося масштабування мережі;

— надати можливість вибору мережі для управління;

— підключення до мережі;

— відображення структури KNX- мережі;

— посилання події до мережі;

— постійне спостереження за подіями які відбуваються в мережі;

— відображення історії користування;

— побудова графіків та діаграм користування елементами мережі;

— відображення подій мережі у реальному часі;

Для досягнення поставленої задачі були сформульовані наступні завдання дослідження:

— провести аналіз проблеми;

— порівняти існуючі рішення;

— виділити основні характеристики подібних систем;

— спроеткувати архітектуру програмного забезпечення;

— розробити програмне забезпечення.

Об’єктом дослідження є структура KNX-мережі.

Предметом дослідження є інструменти для візуалізації структури KNX-мережі, контролю та моніторингу її хостів.

Практичним значенням одержаних результатів є можливість подальшої експлуатації розробленого програмного продукту на підприємствах чи у компанія.

Ключові слова: KNX, ETS, ІНТЕРФЕЙС КОРИСТУВАЧА, МОНІТОРИНГ, ПОДІЯ, МАСШТАБУВАННЯ.

ABSTRACT

The master's dissertation consists of the introduction, 6 main sections, the conclusion, the list of references from 25 titles, 2 applications, 20 figures and 22 tables. The full volume of the master's dissertation is 88 pages, of which the list of links takes 3 pages, and applications - 6 pages.

In the modern world, every person expects more comfort and comfort, both at home and at work. Today KNX is one of the most common solutions for use in medium and large automation systems for houses, offices and commercial premises. The requirements for the convenience, safety and flexibility of modern buildings are constantly growing. At the same time, taking into account the constant technological development and increasing the automation of the systems of comfort, the most important issue is the saving of electricity. That is why the demand for intelligent systems that meet these conditions is constantly increasing.

Using KNX technology can provide solutions that, with the use of conventional methods for creating such systems can only be implemented with difficulty. Control of all applications in an apartment or building can be from one touch panel. From the systems of heating, ventilation and access control, and ending with remote control of all household appliances - KNX opens absolutely new ways to increase comfort, safety and economical energy consumption in apartments and buildings.

The purpose of the thesis is to develop a programmatic implementation of the human-machine interface for visualizing the structure of KNX-network, monitoring and monitoring its hosts.

Assigning software visualization of the structure of various KNX networks, managing its elements and monitoring data for further analysis and identifying more optimal ways of using the network. The software product should provide the following capabilities:

— creating projects for use;
— editing projects in case of network scaling;
— provide the opportunity to choose a network for management;

— network connection;
— display of KNX network structure;
— events event link to the network;
— continuous monitoring of events occurring on the network;
— display of usage history;
— construction of graphs and diagrams of use of elements of the network;
— real-time network events display;

To achieve the task, the following research objectives were formulated:
— to analyze the problem;
— compare existing solutions;
— to highlight the main characteristics of such systems;
— to design the software architecture;
— develop software.

The object of the research is the structure of KNX-network.

The subject of the study is the tools for visualizing the structure of the KNX network, monitoring and monitoring its hosts.

The practical value of the results obtained is the possibility of further exploitation of the developed software product at the enterprises or the company.

Keywords: KNX, ETS, USER INTERFACE, MONITORING, EVENT, EVALUATION.

ЗМІСТ

ВСТУП

В сучасному світі кожна людина очікує більше зручності та комфорту, як у себе вдома так і на роботі. Сьогодні KNX є одним з найбільш поширених рішень для використання в середніх і великих системах автоматизації будинків, офісів і комерційних приміщень. Вимоги до зручності, безпеки та гнучкості сучасних будівель постійно зростають [1]. При цьому, враховуючи постійний технологічний розвиток та підвищення автоматизації систем забезпечення комфорту, найбільш важливим питанням стає економія електроенергії. Саме тому попит на інтелектуальні системи, що відповідають цим умовам, постійно зростає.

Використання технології KNX може надати рішення, які при використанні звичайних методів створення подібних систем можуть бути реалізовані лише насилу. Контроль усіх застосувань в квартирі або будівлі може здійснюватися з однієї сенсорної панелі. Починаючи з систем опалювання, вентиляції і контролю доступу, і закінчуючи дистанційним керуванням усіма побутовими електроприладами - KNX відкриває абсолютно нові шляхи для підвищення комфорту, безпеки і економного енергоспоживання в квартирах і будівлях.

Розроблено людино-машинний інтерфейс для візуалізації структури KNX – мережі, контролю та моніторингу її хостів, який забезпечує можливість візуалізації необмеженої кількості мереж, управління їх елементами та моніторингу подій які відбуваюсь у мережі, що надає можливість проведення досліджень та визначення більш оптимальних шляхів використання KNX-мереж. Веб-додаток розрахований як для інженерів-проектувальників KNX-мереж так і для користувачів які не є спеціалістами у даній області.

Для розробки інтерфейсу веб-додатку було використано JavaScript фреймворк Angular. Для реалізації серверної частини програмного продукту використано Node.js та систему управління базами даних MongoDB. Середовищем реалізації програмного

продукту було обрано WebStorm від компанії .

У першому розділі “Постановка завдання магістерської роботи” описано мету та призначення системи.

У другому розділі “Методи та засоби реалізації візуалізації структури KNX-мереж” описується KNX стандарти та програмний інструмент для проектування KNX-мереж EST.

У третьому розділі “Опис інструментів програмної реалізації” аналізуються інструменти, здатні вирішити поставлену мету та програмно реалізувати призначення системи.

Четвертий розділ “Опис програмної реалізації системи” описує архітектуру системи та принципи взаємодії її компонентів.

У п’ятому розділі “Методи роботи користувача” описано системні вимоги до програмного продукту, шляхи його інсталяції, розглянуто сценарій роботи користувача з веб-додатком.

Шостий розділ “Розробка стартап проекту” присвячений аналізу можливостей просування розробленого веб-додатку як стартап проект.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

JS	- JavaScript
TS	- TypeScrip
ETS	- Software Engineering Tool / Інструментальний програмний пакет
ФРЕЙМВОРК	- інфраструктура програмних рішень, що полегшує розробку складних систем
ООП	- об’ектно – орієнтоване програмування
НТУУ “КПІ”	**-** Національний технічний університет України "Київський політехнічний інститут"

1. ПОСТАНОВКА ЗАВДАННЯ МАГІСТЕРСЬКОЇ РОБОТИ

Система повинна являти собою ресурс спрямований на візуалізацію структури KNX-мережі та моніторингу її компонентів.

1.1. Мета створення та призначення системи

Метою дипломної роботи є розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів.

Оскільки KNX є одним з найбільш поширених рішень для використання в середніх і великих системах автоматизації будинків, офісів і комерційних приміщень, система повинна забезпечити можливість контролю компонентів мережі на будь якій відстані за допомогою мережі Інтернет [1].

Враховуючи те, що KNX-мережа може бути легко пристосована до виконання нових завдань і може бути легко розширена, а нові компоненти можна з легкістю підключати до вже працюючої системи, програмний продукт повинен забезпечити можливість імпорту файлів з програмного застосунку ETS, який використовується для проектування KNX-мереж.

Важливою складовою програмного застосунку є зберігання усіх подій які відбуваються у мережі на основі яких можна відобразити історію користування системою та провести дослідження та визначити більш оптимальні шляхи використання KNX-мережі.

При розробці системи потрібно зауважити та забезпечити коректну роботу як з великих настільних комп’ютерів, так і з мобільних пристроїв, та мати адаптивний інтерфейс який легко адаптується під різні розміри екранів.

Програмний продукт повинен забезпечити наступні можливості:

— створення проектів для використання;

— редагування проектів у випадку коли відбулося масштабування мережі;
— надати можливість вибору мережі для управління;
— підключення до мережі;
— відображення структури KNX- мережі;
— посилання події до мережі;
— постійне спостереження за подіями які відбуваються в мережі;
— відображення історії користування;
— побудова графіків та діаграм користування елементами мережі;
— відображення подій мережі у реальному часі;

Висновки до розділу 1

Задача розробки людино машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів. система повинна забезпечити можливість контролю компонентів мережі на будь якій відстані за допомогою мережі Інтернет. Важливою складовою системи є моніторинг події в мережі. Система повинна забезпечити коректну роботу як з великих настільних комп’ютерів, так і з мобільних пристроїв

2. МЕТОДИ ТА ЗАСОБИ РЕАЛІЗАЦІЇ ВІЗУАЛІЗАЦІЇ СТРУКТУРИ KNX-МЕРЕЖІ

Сьогодні KNX є одним з найбільш поширених рішень для використання в середніх і великих системах автоматизації будинків, офісів і комерційних приміщень. Вимоги до зручності, безпеки та гнучкості сучасних будівель постійно зростають. Саме тому попит на інтелектуальні системи, що відповідають цим умовам, постійно зростає.

Інтерфейс користувача — це зовнішня оболонка будь якої програмної системи, що дозволяє користувачу взаємодіяти із програмою; сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача. У графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості вікон, їхнім розташуванням, сортуванням елементів вікон, гнучкими налаштовуванням як самих вікон, так і окремих їхніх елементів, доступністю багатокористувацьких налаштувань.

2.1. Визначення інтерфейсу користувача

Інтерфейс користувача (ІК) - це сукупність засобів, за допомогою яких користувач взаємодіє з різними пристроями (з комп’ютером або побутовою технікою) або іншим складним інструментарієм (системою). Інтерфейс користувача - це такий різновид інтерфейсів, в якому з одного боку - людина, з іншого - машина (пристрій, програмне забезпечення) [2]. За визначенням Національного банку стандартизованих науково-технічних термінів, інтерфейс користувача - це комплекс апаратних і програмних засобів, що забезпечує взаємодію користувача з комп'ютером. ІК часто розуміють лише як зовнішній вигляд програмного заебзпечення (ПЗ), але таке розуміння є надто вузьким, оскільки саме за допомогою інтерфейсу користувач сприймає програму в цілому та використовує її функціональність.

ІК забезпечує підтримку прийняття рішень у визначеній предметній галузі та визначає порядок використання ПЗ і документації до нього. В дійсності, ІК об'єднує усі елементи і компоненти ПЗ, які здатні впливати на взаємодію користувача з програмним забезпеченням. До таких елементів належать: набір задач, які користувач розв’язує за допомогою ПЗ; використовувана програмним забезпеченням метафора (наприклад, "робочий стіл" у операційній системі Windows); елементи управління ПЗ; навігація між блоками ПЗ; візуальний (і не тільки) дизайн вікон та екранних форм програми та інші складові (рисунок 2.1).

Стиль інтерфейсу користувача - це набір ознак, методів, прийомів діяльності, які характеризують індивідуальність інтерфейсу користувача, а також сукупність прийомів використання інструментів розроблення ПЗ.

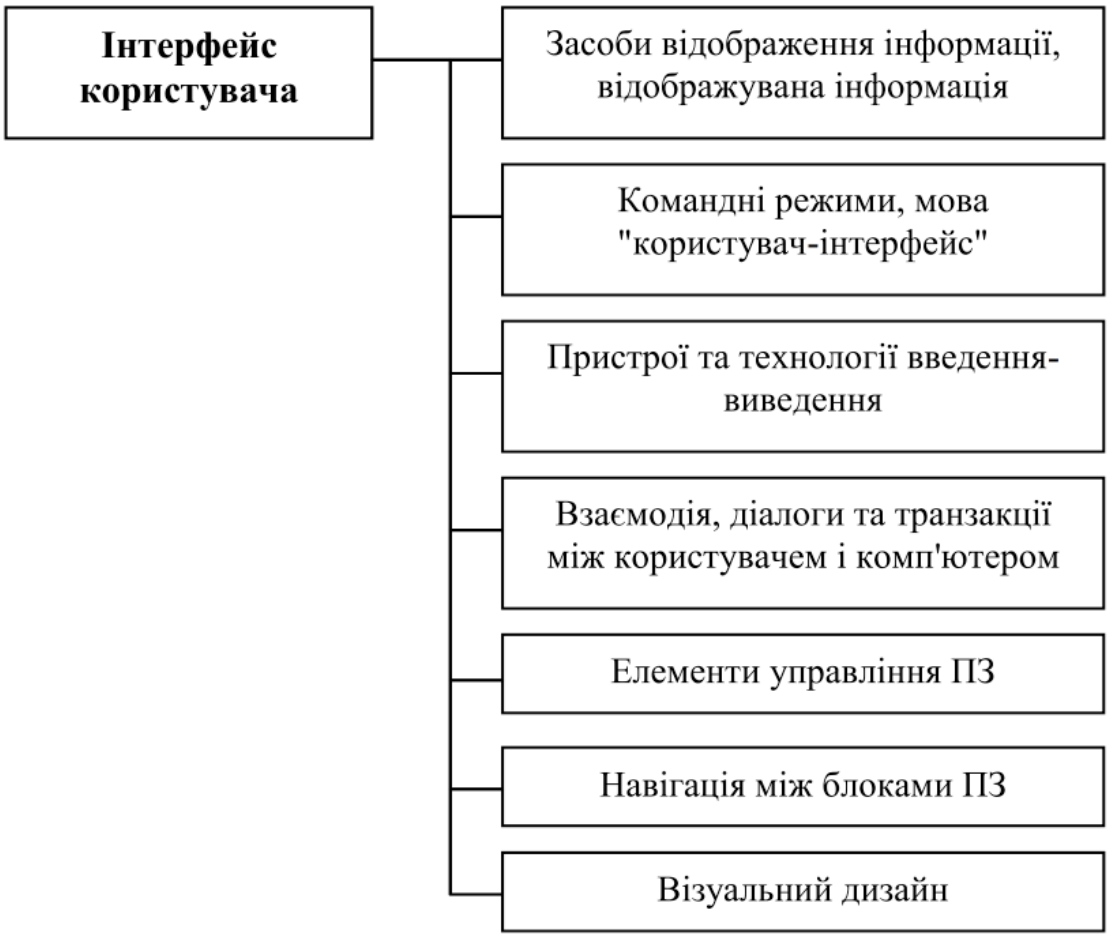

Рис.2.1 - Складові інтерфейсу користувача

Процес проектування ІК – це складний, нелінійний, недетермінований і неортогональний процес. Складність ІК обумовлюється рядом невизначеностей, які суттєво впливають на процес розроблення. Нелінійність проектування ІК полягає у відсутності фіксованого, впорядкованого і прямолінійного алгоритму від початку до кінця проектування. Процес проектування є невизначеним, оскільки не існує рівняння, за яким можна було б одержати однаковий результат при заданих однакових початкових умовах, більш того, одержати ідентичний результат практично неможливо. Інтерфейс користувача неортогональний у тому сенсі, що будь-який аспект проектного рішення може впливати на інші аспекти, до того ж результат цього впливу не завжди є позитивним та прийнятним.

2.2. Проблематика проектування інтерфейсу користувача

Процес проектування сучасного ПЗ передбачає вирішення ряду задач, зокрема: зниження витрат на проектування, скорочення термінів проектування, покращення якості пропонованих рішень, забезпечення нескладного в освоєнні та використанні ПЗ, вивчення та впровадження нових технологій та засобів, досягнення кращих результатів в порівнянні з конкурентами.

Задоволеність користувача програмним продуктом або зручністю його використання в значній мірі визначається інтерфейсом користувача. Взагалі, задоволеність користувача – це функція невеликої кількост факторів[2]:

— можливостей інтерфейсу користувача;
— час відгуку;
— надійність;
— пристосованість до інсталяції;
— інформаційна підтримка;
— пристосованість до супроводження

Виділені прописними літерами фактори відіграють найбільш важливу роль. Можливості ІК повинні повністю відображати функціональні можливості програми. Час відгуку ІК повинен бути мінімальним, щоб користувачу не доводилось довше, ніж потрібно, очікувати виконання заданої дії. Надійність ІК - це властивість зберігати в часі у встановлених межах значення всіх параметрів, які характеризують здатність виконувати потрібні функції в заданих режимах та умовах застосування. Пристосованість до інсталяції відіграє суттєву роль у задоволеності користувача, оскільки користувач починає своє знайомство з програмним засобом та його інтерфейсом саме з процесу інсталяції програмного засобу. Інформаційна підтримка користувача - це навчальна та довідкова складові ПЗ, від яких залежить, наскільки швидко і легко користувач опанує новий програмний продукт. Супроводження ІК - це процес покращення, оптимізації та усунення недоліків ІК після передачі програмного забезпечення в експлуатацію. Пристосованість ІК до супроводження важлива можливістю покращення ІК вже за участі користувачів. До інших факторів відносяться узгодженість, інтегрованість та вартість ІК, які впливають на задоволеність користувача інтерфейсом, а відтак і програмним продуктом в цілому.

Усі фактори задоволеності користувача та їх відносну важливість слід враховувати під час кожного етапу життєвого циклу програмного забезпечення ІК.

До труднощів проектування ІК слід віднести і той факт, що користувач не завжди може чітко висловити свої вимоги та побажання щодо програмного продукту та його інтерфейсу на етапі проектування, але є дуже категоричним щодо бажаності і небажаності тих чи інших властивостей на етапі введення ПЗ в експлуатацію[2].

Часто характеристики ІК щодо практичності, інтеграції та узгодженості не формулюються явно на етапі проектування ПЗ, а визначаються на рівні деяких очікувань, що призводить до невірного розуміння очікувань замовника проектувальниками ПЗ. Тому такі вимоги повинні визначатись явно, причому бути вимірюваними (мати кількісні характеристики), оскільки проектувальник ІК може не бачити і не розуміти видимі та зрозумілі замовникам і користувачам вимоги.

2.3. Принципи проектування інтерфейсу користувача

Взаємодія між користувачем і комп’ютером (HCI - Human-Computer Interaction) відбувається в інтерфейсі. Основною метою HCI є покращення взаємодії між користувачем і комп’ютером, роблячи комп’ютери більш кориснішими і сприйнятливими до потреб користувачів.

Найчастіше ефективність використання всіх функцій системи й ефективність роботи самої системи визначається у більшому ступені тим, як побудований її інтерфейс. [3].

Природний інтерфейс — такий інтерфейс, що не потребує від користувача істотно змінювати звичні для нього способи розв’язання задачі. Це, зокрема, означає, що повідомлення і результати, що отримуються за допомогою програмного продукту, не повинні вимагати додаткових пояснень. Доцільно також зберігати систему позначень і термінологію, які використовуються в даній предметній галузі.

Використання знайомих користувачу понять і образів (метафор) забезпечує інтуїтивно зрозумілий інтерфейс при виконанні завдань.

Метафори є свого роду «містком», що зв'язує образи реального світу з тими діями і об'єктами, якими доводиться маніпулювати користувачу при його роботі на комп'ютері; вони забезпечують «пізнавання», а не «згадку». Користувачі запам'ятовують дію, пов'язану із знайомим об'єктом, легше, ніж вони запам'ятали б ім'я команди, пов'язаної з цією дією.

Узгодженість інтерфейсу означає використання однакових або дуже схожих метафор і способів взаємодії з користувачем і порядку роботи в різних додатках. Узгодженість надає можливість користувачам переносити наявні знання на нові завдання, освоювати нові аспекти швидше, і завдяки цьому фокусувати увагу на задачу, що вирішується, а не витрачати час на з'ясування відмінностей у використовуванні тих

або інших елементів управління, команд і т.д. Забезпечуючи спадкоємність одержаних раніше знань і навичок, узгодженість робить інтерфейс зрозумілим і передбачуваним.

Узгодженість важлива для всіх аспектів інтерфейсу, включаючи імена команд, візуальне представлення інформації і поведінку інтерактивних елементів.

Узгодженість в межах продукту

Одна і та ж команда повинна виконувати одні і ті ж функції, де б вона не зустрілася, причому однаково.

Узгодженість в межах робочого середовища

Підтримуючи узгодженість з інтерфейсом, що надається операційною системою (наприклад, ОС Windows), програмний продукт може «спиратися» на ті знання і навички користувача, які він одержав раніше при роботі з іншими ПП.

Узгодженість у використовуванні метафор

Якщо поведінка деякого програмного об'єкту виходить за рамки того, що звичайно мається на увазі під відповідною йому метафорою, у користувача можуть виникнути труднощі при роботі з таким об'єктом. Наприклад, якщо для програмного об'єкту Корзина визначити операцію Запуск, то для з'ясування її значення користувачу, швидше за все, знадобиться стороння допомога.

Користувачі звичайно вивчають особливості роботи з новим програмним продуктом методом спроб і помилок. Ефективний інтерфейс повинен брати до уваги такий підхід: на кожному етапі роботи він повинен надавати можлівість тільки відповідний набір дій і попереджати користувачів про ті ситуації, де вони можуть нашкодити Системі або даним; ще краще, якщо у користувача існує можливість відмінити або виправити виконані дії.

Гнучкість інтерфейсу — це його здатність враховувати рівень підготовки і продуктивність праці користувача. Властивість гнучкості припускає можливість зміни структури діалогу та вхідних даних[3]. Концепція гнучкого (адаптивного) інтерфейсу в даний час є однією з основних областей дослідження взаємодії людини і комп’ютера.

Основна проблема полягає не в тому, як організувати зміни в діалозі, а в тому, які ознаки потрібно використовувати для визначення необхідності внесення змін і їх суті.

Проектування візуальних компонентів є найважливішою складовою частиною розробки програмного інтерфейсу. Коректне візуальне представлення використовуваних об'єктів забезпечує передачу досить важливої додаткової інформації про поведінку і взаємодію різних об'єктів. У той же час слід пам'ятати, що кожен візуальний елемент, який з'являється на екрані, потенційно вимагає уваги користувача, яка, як відомо, не безмежна. Необхідно забезпечити формування на екрані такого середовища, яке не тільки сприяло б розумінню користувачем представленої інформації, але і дозволяло б зосередитися на найважливіших її аспектах.

Принцип «Гаманець Міллера» названий так на честь вченого-психолога Г. А. Міллера, який досліджував короткотривалу пам'ять, перевіряючи висновки, зроблені раніше його колегою, Г. Еббінгаузом. Еббінгауз намагався з'ясувати, скільки інформації може запам'ятати людина без будь-яких спеціальних мнемонічних прийомів.Виявилося, що ємність пам'яті обмежена сім'ю цифрами, сім'ю літерами або назвами семи предметів. Це "магічне число" сім, що служить свого роду міркою пам'яті, і було перевірено Міллером, який показав, що пам'ять дійсно в середньому не може зберігати більше семи елементів; в залежності від складності елементів це число може коливатися в межах від п'яти до дев'яти.

Якщо необхідно протягом короткого часу зберегти інформацію, що включає більше семи елементів, мозок майже несвідомо групує цю інформацію таким чином, щоб число запам'ятовуються елементів не перевищувало гранично допустимого. Наприклад, номер банківського рахунку 30637402710, що складається з одинадцяти елементів, буде, швидше за все, запам'ятовуватися як 30 63 740 27 жовтня, тобто як п'ять числових елементів, або вісім слів (тридцять, шістдесят, три, сімсот, сорок , двадцять, сім, десять)[4].

Застосовуючи принцип гаманця Міллера в дизайні інтерфейсів, слід групувати елементи в програмі (кнопки на панелях інструментів, пункти меню, закладки, опції на

цих закладках і т. п.) з урахуванням цього правила-тобто не більше семи в групі, в крайньому випадку - дев'яти. Погляньте, наприклад, на головне вікно програми-словника ABBYY Lingvo 6.0: чотирнадцять кнопок на верхній панелі, між якими немає жодного роздільника, сприймаються набагато гірше, ніж кнопки на панелі внизу, які розділені на групи.

Отже, принцип гаманця Міллера каже про сім плюс-мінус двох елементах. Але якщо поглянути на програми, інтерфейс яких удосконалювався роками (той же Microsoft Word), то можна помітити, що число об'єктів (пунктів меню, кнопок на панелях інструментів) в групах доходить до шести-семи досить рідко, а в основному елементи згруповані по три-чотири об'єкта. Такі невеликі групи об'єктів найбільш добре сприймаються поглядом користувача, вже трохи втомленого складними інтерфейсами сучасних програм. Я думаю, при проектуванні інтерфейсів програм верхню межу гаманця Міллера - сім-дев'ять елементів - потрібно застосовувати дуже обережно, намагаючись обходитися групами, що містять максимум п'ять об'єктів.

2.4. Комунікаційна шина KNX

У сучасних системах автоматизації будівель використовується цілий ряд різних стандартів зв'язку, і KNX займає серед них почесне місце. Ця розробка рідко зустрічається в сегменті DIY-рішень, але зате в професійних системах вона застосовується вже давно, причому, вельми успішно.

KNX - комунікаційна шина, широко використовувана для автоматизації будівель. Стандарт шини KNX, став розвитком більш ранньої розробки EIB (аббр. Від англ. European Installation Bus, рус. Європейська інсталяційна шина)[5]. EIB - застаріле позначення, але воно продовжує використовуватися, особливо в Європі. Іноді використовується позначення EIB / KNX. Продукція KNX поширювалася під декількома

торговельними марками. Найбільш відомі GIRA, JUNG, THEBEN, INSTABUS, ABB i-Bus, TEBIS, INTESIS BOX, SCHNEIDER.

У KNX багате минуле. Почнемо з того, що в основі стандарту лежить шина EIB (European Installation Bus), яка з'явилася ще на початку 90-х років. Основні принципи, які використовуються в роботі KNX, були сформовані вже тоді. До кінця століття EIB була явним лідером у своїй галузі, однак у неї були гідні конкуренти. Йдеться про шину Batibus, що набула поширення на півдні Європи, а також про стандарт EHS (European Home System), що сподобався виробникам побутової техніки. Три консорціуму, відповідальних за просування EIB, Batibus і EHS, вирішили об'єднати зусилля для розробки нового, більш досконалого рішення. В результаті в 1999 році на світ з'явилася «Асоціація KNX».

Рисунок 2.2 — Комунікаційна шина KNX

Сам стандарт KNX був представлений навесні 2002 року. Як виявилося, близько 80 відсотків розробок, що лежать в основі новинки, була запозичена у EIB. Від двох інших «донорів» дебютанту дісталися механізми завдання налаштувань і нові способи передачі сигналів. З цієї причини, EIB і KNX часто прирівнюють, нерідко шину називають «EIB / KNX»[5]. В кінці 2003 року розробка була оформлена як європейського стандарту EN 50090, а ще через три роки вона набула статусу міжнародного стандарту ISO / IEC 14543. Іншими словами, KNX успішно застосовується і за межами Європи.

У чому переваги стандарту? В першу чергу, він славиться своєю надійністю: незважаючи на наявність відразу кількох середовищ передачі даних, основні компоненти KNX-систем зазвичай об'єднуються воєдино за допомогою спеціальних кабелів, причому в системі передбачений механізм підтвердження отримання пакетів, тобто якщо команда не дійшла до мети, то вона відправляється повторно (не більше двох разів). Інший очевидний плюс дротова мережа - можливість розміщення обладнання на значній відстані один від одного. Також слід зазначити, що KNX-пристрої не відчувають проблем з сумісністю, чого не можна сказати, наприклад, про конкурентну продукцію. Ще один аргумент на користь KNX - гнучка масштабованість. Стандарт можна однаково успішно використовувати як в приватних будинках, так і лікарнях або аеропортах.

Недоліки у KNX теж є: даний продукт орієнтований на професійні системи автоматизації, проектуванням і установкою яких займаються компанії-інтегратори. Самостійний монтаж KNX-мережі є досить складним завданням.

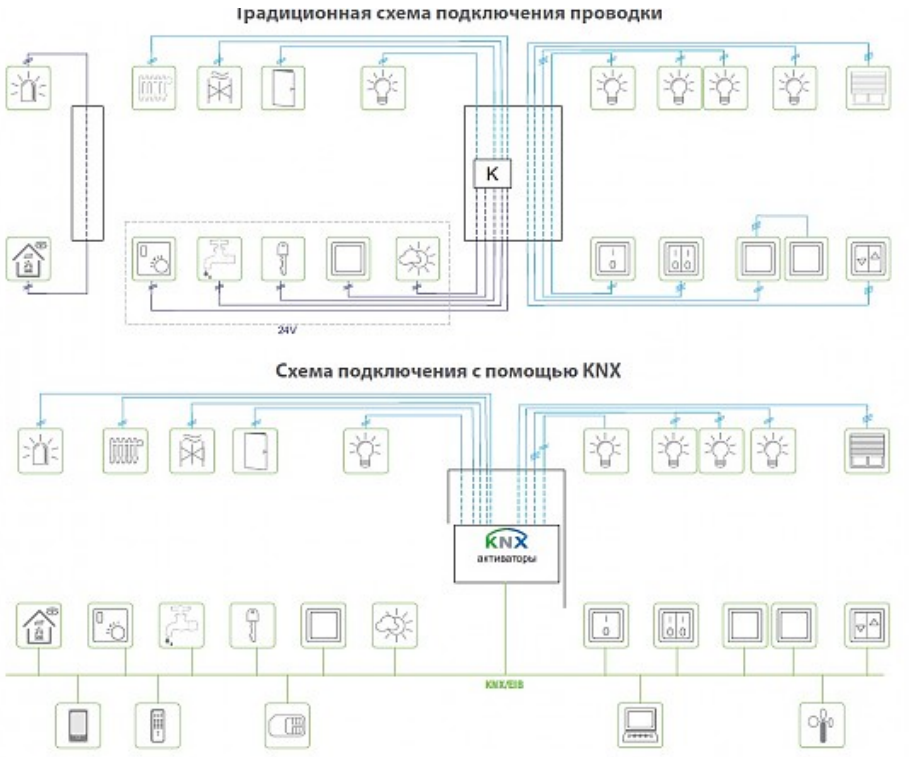

Рисунок 2.3 — Схема порівняння підключення

Компоненти KNX-мережі можна розділити на три основні групи. Перша з них складається з пристроїв, що генерують керуючі команди. Це можуть бути вимикачі, контрольні панелі, різні сенсори і таймери, а також датчики вимірювання фізичних величин. Друга категорія - це актуатори, тобто виконавчі пристрої. В їх число входять релейні модулі і різні регулятори - наприклад, перемекачі. Третю групу утворюють допоміжні системні пристрої, такі як блоки живлення, повторювачі, логічні модулі і інтерфейси, що забезпечують зв'язок із зовнішнім світом. Слід зазначити, що мережа KNX є децентралізованою: сенсори і актуатори можуть обмінюватися даними безпосередньо, без участі додаткового контролера.

Стандарт KNX передбачає відразу чотири середовища передачі даних: окрема шина (кручена пари), електропроводка, радіоканал і IP-мережу. Не можна сказати, що вони рівноправні. Шинне з'єднання дозволяє використовувати різні варіанти топології мережі і об'єднувати велику кількість пристроїв, що знаходяться на значній відстані один від одного.

У найпростішому варіанті дротова мережа KNX є сегмент з топологією «лінія». Він може включати в себе до 64 шинних пристроїв. Максимальна довжина однієї лінії становить кілометр, але за допомогою спеціальних підсилювачів це значення можна збільшити в чотири рази. Кожен сегмент зобов'язаний мати власний блок живлення. До 15 лінії можуть бути підключені до так званої головної лінії і об'єднані тим самим в «зону». У свою чергу, зони (до 15 штук) вміють спілкуватися між собою за допомогою магістральної лінії.

Швидкість передачі даних всередині KNX-мережі становить близько 9600 біт / с, але за рахунок малого обсягу трансльованих повідомлень (кілька байт) цього вистачає для забезпечення гарної чуйності: середній час відгуку на команду становить всього 25 мс. Для переданих пакетів задається пріоритет. Слід зазначити, що в такій мережі використовується відразу два види адрес - фізичні та групові. Останній варіант використовується в тих випадках, коли одну команду необхідно відіслати відразу декількох пристроїв - він визначає приналежність гаджета до тієї чи іншої умовної групи. Фізична адреса у елемента мережі завжди один, а ось групових може бути кілька.

Бездротова версія KNX використовує частоту 868 МГц, при цьому на передачу сигналу від окремих пристроїв витрачається не більше одного відсотка ефірного часу, що дозволяє уникнути тривалих перешкод, які блокують радіоканал. Максимальна швидкість передачі даних - приблизно 16 400 біт / с. Гаджети з односпрямованої зв'язком відсилають пакети негайно, двонаправлені варіанти попередньо перевіряють, чи вільний радіоканал. Повідомлення, що передаються «по повітрю», крім усього іншого, містять такі дані, як рівень заряду батареї і серійний номер пристрою. Останній дозволяє уникнути проблем при використанні декількох радіомереж на одній території. За радіусу покриття KNX можна порівняти зі своїми прямими конкурентами, при цьому далекобійність на окремих ділянках може бути збільшена за допомогою спеціальних повторювачів.

Можливість використання електропроводки в якості середовища передбачена для тих випадків, коли прокладка нового кабелю утруднена, а радіосигнал не поширюється

на достатню відстань. Технологія дозволяє домогтися швидкостей близько 1200 біт / с. Слід зазначити, що цей варіант не користується популярністю, і в практичних реалізаціях він зустрічається рідко. Що стосується такого середовища, як IP-мережу, то вона застосовується для тунелювання і маршрутизації KNX-команд шляхом їх перетворення в IP-пакети. Дана можливість особливо корисна в тих випадках, коли взаємодіючі пристрої розташовуються на значній відстані один від одного.

2.5. Програмний інструмент ETS

ETS (Software Engineering Tool / Інструментальний програмний пакет) - це унікальний, універсальний інструментальний програмний пакет для розробки та налаштування "розумної" KNX системи автоматизованого управління та контролю будівель та квартир [6].

Асоціація KNX як засновник і власник KNX Standard пропонує ETS як інструмент налаштування пристроїв, який фактично є частиною стандарту KNX і, отже, є частиною системи KNX. ETS - це по-справжньому унікальний програмний пакет, у будь-якій точці світу ви можете використовувати одну і ту ж програмну пакету ETS для будь-якого проекту на базі KNX стандарту і для будь-якого пристрою, сертифікованого Асоціацією KNX. Даний програмний пакет є дійсною частиною стандарту KNX. В ETS можуть бути імпортовані будь-які бази даних KNX продуктів всіх виробників, що входять в Асоціацію KNX! Система KNX в поєднанні програмним пакетом ETS представляє собою легкий у використанні і не залежить від виробника інструментарій для проектувальників систем управління і інсталяторів.

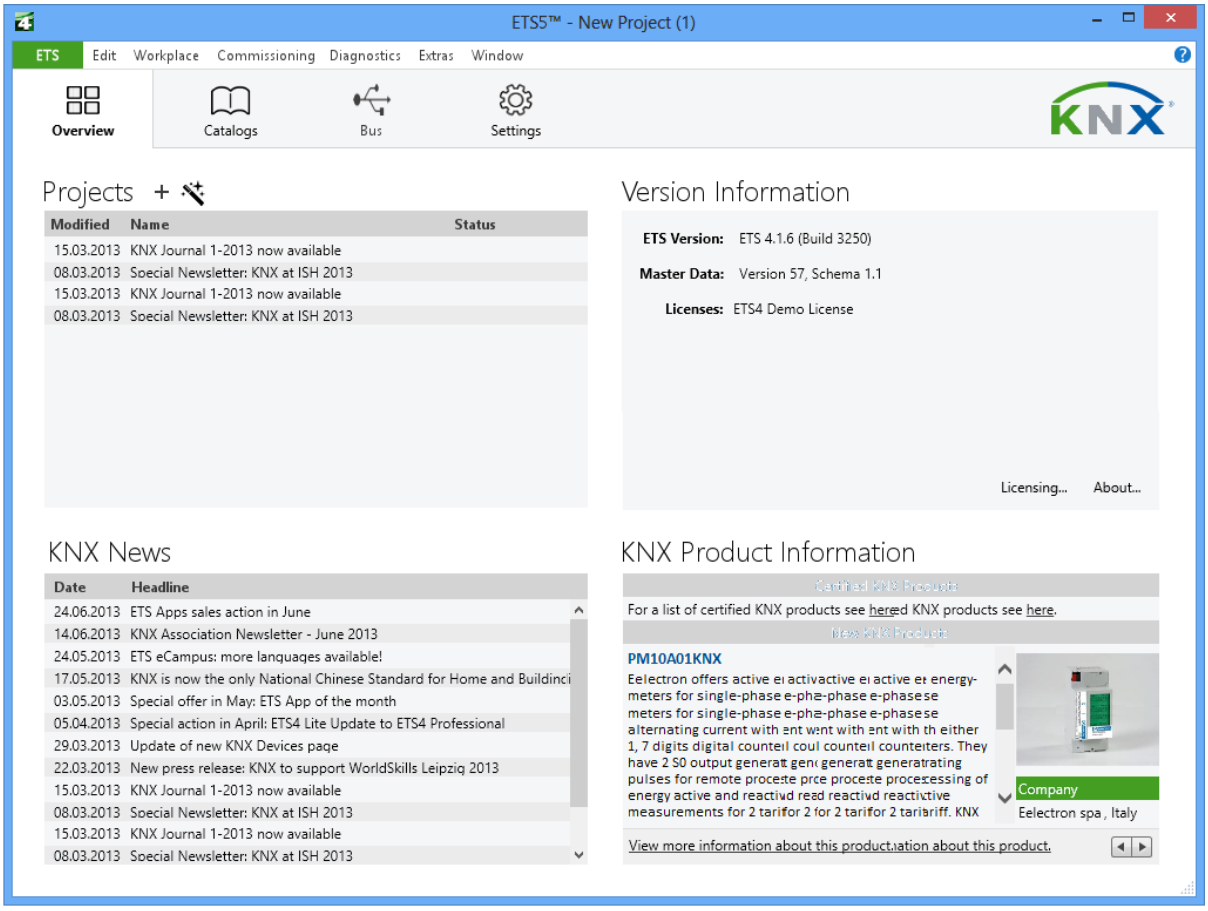

Рисунок 2.4 — Програмний інтерфейс ETS

Основні переваги:

– гарантія максимальної сумісності програмного забезпечення ETS зі стандартом KNX;

– всі апплікаційні дані сертифікованих пристроїв усіх виробниківотримання інформації із Всесвітнв KNX можуть бути імпортовані в ETS;

– імпорт та експорт проектів;

– всюду в світі всі дизайнеры та установники використовують один і той же інструмент ETS для кожного проекту KNX і з кожним сертифікованим KNX пристроєм. Гарантується надійний обмін даними;

ETS - це програмне забезпечення, яке працює на комп'ютерах під операційною системою Windows

2.6. Структура *.esf файлу

ETS надає можливість експорту проекту у форматі .esf, файл містить підмножину доступних даних проекту ETS, це обмеження застосовується як до типів даних, так і до структури групових адресів [7].

У першому рядку вказується назва проекту ETS. Зазначені групові адреси проекту ETS перелічені з другої лінії на наступний, відповідно формат для такої лінії виглядає наступним чином(рисунок 2.4).

```
Project 2018 EN Final - Empty
Ground floor.Lighting.0/0/1 Switching Living Room E13   EIS 1 'Switching' (1 Bit)   Low 2/0/0
Ground floor.Lighting.0/0/2 Status Living Room E13  Uncertain (1 Byte)  Low
Ground floor.Lighting.0/0/3 Relative Dimming Living Room E13    EIS 2 'Dimming - control' (4 Bit)   Low
Ground floor.Lighting.0/0/4 Absolute Dimming value Living Room E13  Uncertain (1 Byte)  Low
Ground floor.Lighting.0/0/5 Status Living Room E13 - Status On/Off  EIS 1 'Switching' (1 Bit)   Low
```

Рисунок 2.5 — Приклад *.esf файлу

Рядки що відповідають за групові адреса мають наступ структуру :

– Main Group Address Name – головна группа;

– Middle Group Address Name -підгрупа;

– Group Address Number - це число, розділене символом "/" (кодований як \ 0x20), вона перераховується лише один раз;

– Group Address Name – назва групи;

– Data Type – тип даних;

– Priority;

– Connected Group Address - груповий адрес який повз’язаний з іншими групами;

Висновки до розділу 2

У другому розділі дано визначення інтерфейсц користувача, розглянено пробематику побудови інтерфейсу та описані основні принципи проектування інтерфейсу користувача. Визначено поняття KNX-мережі особливості її структури та роботи з нею. Наведено приклад підключення елементів за допомогою KNX-мережі та без неї. Розглянуто головний програмний інструмент ETS для розробки та налаштування "розумної" KNX системи автоматизованого управління та контролю будівель та квартир. Опис структури *.esf файлу який експортується з програмного інструменту ETS.

3. ОПИС ІНСТРУМЕНТІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

При розробці програмного продукту важливим чинником є правильний вибір засобів програмної реалізації, що впливає на час розробки, якість, надійність та швидкість роботи продукту. Тому, для розробки програмного продукту використано середовище розробки WebStorm від компанії JetBrains, мову програмування JavaScript. Для реалізації серверної частини програмного продукту використано Node.js та систему управління базами даних MongoDB. Для розробки клієнтської частини продукту використано Angular, HTML для розмітки веб сторінки та CSS для опису веб сторінки. Для створення певних деталей інтерфейсу використовувався Adobe Photoshop CS6, що є програмою для створення та редагування графічних елементів.

3.1 Середовище розробки JetBrains WebStorm

Програмне забезпечення JetBrains WebStorm являє собою інструмент для розробки web-сайтів і редагування HTML, CSS і JavaScript коду. Рішення забезпечує швидку навігацію по файлах і генерує повідомлення про виникаючі проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементів SQL безпосередньо в JavaScript. JetBrains WebStorm здійснює розгортання і синхронізацію проектів через протокол FTP [8].

Використовуючи можливості коду HTML / XHTML і XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів і інших елементів коду. При роботі з CSS здійснюється завершення коду класів, HTML-номерів, ключових слів і т. д.. WebStorm пропонує автоматичне рішення таких проблем, як вибір формату, властивостей, класів, посилань на файли і інших атрибутів CSS. Рішення дозволяє використовувати потужність інструменту Zen coding для верстки HTML, відображає дії тега на web-сторінці. Продукт WebStorm здійснює завершення коду JavaScript для

ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує специфічні особливості популярних браузерів. Реалізовані в рішенні функції рефакторінга JavaScript дозволяють перетворювати структуру коду і файлів і .js.

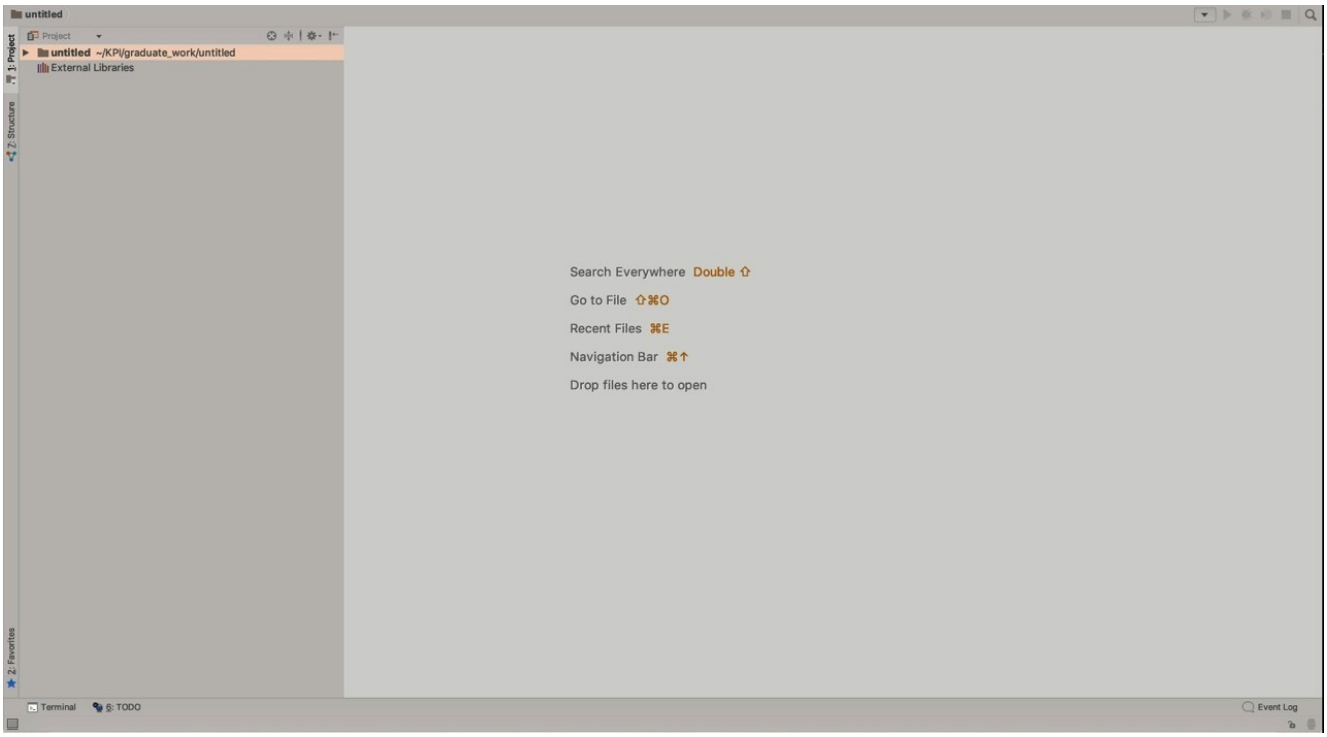

Рисунок 3.1 — Середовище розробки JetBrains WebStorm

WebStorm забезпечує налагодження коду JavaScript і надає широкий діапазон можливостей: знаходження точки зупину в HTML і JavaScript, настройка параметрів точки зупину, тестування синтаксису коду в режимі реального часу і т. д.. Продукт підтримує платформи JQuery, YUI, Prototype, DoJo, MooTools, Qooxdoo і Bindows. WebStorm передбачає інтегровану перевірку тексту на теги, послідовність коду, помилки в написанні і т. Д. WebStorm дозволяє редагувати файли і автоматично синхронізувати їх на вимогу при віддаленій роботі або зберіганні.

WebStorm підтримує новітній стандарт JavaScript, а також TypeScript, пропонуючи автодоповнення коду, перевірку помилок, рефакторингом, форматування і налагодження. WebStorm дозволяє ефективно розробляти програми на Node.js і

підтримує повноцінне налагодження Node.js додатків. Новий додаток можна створити, використовуючи шаблон Node.js Express, а необхідні модулі легко встановити через npm, не використовуючи командний рядок. Live Edit дозволяє миттєво бачити результати ваших змін в CSS, HTML і JavaScript файлах проекту в браузері без перезавантаження сторінки. Live Edit доступний в режимі налагодження JavaScript і працює в браузері Google Chrome.

Ключові можливості:

- налагодження client-side, Node.js і React Native додатків
- допомога при роботі з додатками на Angular, React, Vue.js, Electron і React Native
- інтеграція з системами управління версіями Git, GitHub, Subversion, Perforce і Mercurial
- інтеграція з системами стеження за помилками (ESLint, JSHint, JSLint, TSLint і Stylelint)

Продукт підтримує функцію контролю версій і попередніх варіантів коду і фіксує всі вироблені дії та зміни. Завдяки створенню історії в WebStorm можна відновлювати кодові вирази, блоки і навіть цілі файли.

3.2 Мова програмування JavaScript

JavaScript - динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше вікорістовується як частина браузера, що надає можлівість коду на стороні клієнта (такого, що віконується на пристрої кінцевого користувача) взаємодіяті з користувачем [9], Керувати браузером, асинхронно обмінюватіся данімі з сервером, змінюваті структуру та Зовнішній вигляд веб-сторінки.

Мова JavaScript також вікорістовується для програмування на стороні сервера (подібно до таких мов програмування, як Java і C #), розробки ігор, стаціонарних та

мобільніх Додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), Всередині PDF-документів тощо.

JavaScript класіфікують як прототипна (підмножіна об'єктно-орієнтованої), скриптовими мову програмування з дінамічною тіпізацією.

Окрім прототіпної, JavaScript також частково підтрімує інші парадигми програмування (імперативну та частково функціональну) и деякі відповідні архітектурні Властивості, зокрема: динамічна та Слабкий тіпізація, Автоматичне керування пам'яттю, прототипна наслідування, Функції як об'єкти первого класу.

3.3 Строго типізований TypeScript

Розвиток TypeScript розпочався в кінці 2012 року. Хоча він зародився в компанії Microsoft, і його фактичним творцем є програміст Андерс Хейлсберг, так само відомий як творець таких мов як Delphi, C #, але даний проект відразу став розвиватися як OpenSource. І вже з самого початку нова мова стала швидко поширюватися в силу своєї гнучкості і продуктивності. Чимало проектів, які були написані на JavaScript, стали переноситися на TypeScript. Популярність і актуальність ідей нової мови привела до того, що ряд з цих ідей в подальшому стануть частиною нового стандарту JavaScript. А нова версія одного з найпопулярніших фреймворків для Web - Angular 2/4/5/6 повністю написана на TypeScript спільно компаніями Microsoft і Google[10].

Однак, здавалося б, навіщо потрібна ще одна мова програмування для клієнтської сторони в середовищі Web, якщо з усією тією ж самою роботою прекрасно справляється і традиційний JavaScript, який використовується практично на кожному сайті, яким володіє безліч розробників і підтримка якого є досить висока . Але TypeScript це не просто новий JavaScript.

По-перше, слід зазначити, що TypeScript - це строго типізована і компільована мова, ніж, можливо, буде ближче до таких мов програмування як Java, C # та інших строго типізованих мов. Хоча на виході компілятор створює все той же JavaScript, який потім виконується браузером. Однак сувора типізація зменшує кількість потенційних помилок, які могли б виникнути при розробці на JavaScript.

По-друге, TypeScript реалізує багато концепцій, які властиві об'єктно-орієнтованим мовам програмування, наприклад: успадкування, поліморфізм, інкапсуляція, модифікатори доступи та інші.

По-третє, потенціал TypeScript дозволяє швидше та простіше писати великі та складні комплексні програми, відповідно їх легше підтримувати, розвивати, маштабувати і тестувати, ніж на стандартному JavaScript[11].

По-четверте, TypeScript розвивається як opensource-проект і, як і багато проектів, хоститься на гітхабі. Крім того, він є кросплатформеним, а це означає, що для розробки ми можемо використовувати як Windows так і MacOS або Linux.

У той же час TypeScript є надмножиною JavaScript, а це означає, що будь-яка програма на JS є програмою на TypeScript. В TS можна використовувати всі ті конструкції, які застосовуються в JS – ті ж оператори, умовні та циклічні конструкції. В кінцевому рахунку, TS - це всього лише інструмент, який покликаний полегшити розробку додатків.

Згенерований компілятором TypeScript код в JS підтримується переважною більшістю браузерів, так як орієнтується насамперед на стандарт ECMAScript 3, хоча TS також підтримує і стандарти ECMAScript 5 і ECMAScript 2015 / 2017. Хоча в процесі розробки ми можемо самі задати цільової стандарт ECMAScript.

3.4 Серверна платформа Node.js

Node.js - це кросплатформенне седеродище виконная JavaScript коду , яка працює на серверах. З моменту випуску цієї платформи в 2009 році вона стала надзвичайно популярною і в наші дні грає дуже важливу роль в області веб-розробки. Якщо вважати показником популярності число зірок, які зібрав якийсь проект на GitHub, то Node.js, у якого більше 50000 зірок, це дуже і дуже популярний проект[12].

Платформа Node.js побудована на базі JavaScript движка V8 від Google, який використовується в браузері Google Chrome. Дана платформа, в основному, використовується для створення веб-серверів, проте сфера її застосування цим не обмежується.

Однією з основних привабливих особливостей Node.js є швидкість. JavaScript-код, що виконується в середовищі Node.js, може бути в два рази швидше, ніж код, написаний на компільованих мовах, на приклад C або Java, і на порядки швидше інтерпретованих мов на зразок Python або Ruby. Причиною подібного є не блокуюча архітектура платформи, а конкретні результати залежать від використаних тестів продуктивності, але, в цілому, Node.js - це дуже швидка платформа.

Платформа Node.js проста в освоєнні і використанні. Насправді, вона прямо-таки дуже проста, особливо це помітно в порівнянні з деякими іншими серверними платформами.

У середовищі Node.js виконується код, написаний на JavaScript. Це означає, що мільйони фронтенд-розробників, які вже користуються JavaScript в браузері, можуть писати і серверний, і клієнтський код за допомогою однієї мови програмування без необхідності вивчати абсолютно нових інструментів для переходу до серверної розробки.

У браузері і на сервері використовуються однакові концепції мови. Крім того, в Node.js можна оперативно переходити на використання нових стандартів ECMAScript у міру їх реалізації на платформі. Для цього не потрібно чекати до тих пір, поки

користувачі оновлять браузери, так як Node.js - це серверне середовище, яке повністю контролює розробник. В результаті нові можливості мови виявляються доступними при установці необхідної версії Node.js.

В основі Node.js, крім інших рішень, лежить JavaScript-движок V8 від Google, який застосовується в браузері Google Chrome і в інших браузерах. Це означає, що Node.js користується напрацюваннями тисяч інженерів, які зробили середу виконання JavaScript неймовірно швидкої і продовжують працювати в напрямку вдосконалення V8.

У традиційних мовах програмування (C, Java, Python, PHP) всі інструкції, за замовчуванням, є блокуючими, якщо тільки розробник явно не подбає про асинхронне виконання коду. В результаті якщо, в такому середовищі, зробити мережевий запит для завантаження якогось JSON-коду, виконання потоку, з якого зроблено запит, буде призупинено до тих пір, поки не завершиться отримання і обробка відповіді.

JavaScript значно спрощує написання асинхронного і неблокуючим коду з використанням єдиного потоку, функцій зворотного виклику (коллбеков) і підходу до розробки, заснованої на події. Кожен раз, коли нам потрібно виконати важку операцію, ми передаємо відповідним механізмам коллбек, який буде викликаний відразу після завершення цієї операції. В результаті, для того щоб програма продовжила роботу, чекати результатів виконання подібних операцій не потрібно.

Подібний механізм виник в браузерах. Ми не можемо дозволити собі чекати, скажімо, закінчення виконання AJAX-запиту, не маючи при цьому можливості реагувати на дії користувача, наприклад, на клацання по кнопках. Для того щоб користувачам було зручно працювати з веб-сторінками, все, і завантаження даних з мережі, і обробка натискання на кнопки, має відбуватися одночасно, в режимі реального часу.

Асинхронні механізми дозволяють єдиному Node.js-сервера одночасно обробляти тисячі підключень, не навантажуючи при цьому програміста завданнями з управління потоками і по організації паралельного виконання коду. Подібні речі часто є джерелами помилок.

Node.js надає розробнику не блокуючі базові механізми введення, виведення і бібліотеки, які використовуються в середовищі Node.js, написані з використанням не блокуючих парадигм. Це робить блокуючу поведінку коду швидше винятком, ніж нормою.

Коли Node.js потрібно виконати операцію вводу-виводу, на зразок завантаження даних з мережі, доступу до бази даних або до файлової системи, замість того, щоб заблокувати очікуванням результатів такої операції головний потік, Node.js ініціює її виконання і продовжує займатися іншими справами до тих пір, поки результати виконання цієї операції не будуть отримані.

Завдяки простоті і зручності роботи з менеджером пакетів для Node.js, який називається npm, екосистема Node.js прямо-таки процвітає. Зараз в реєстрі npm є понад півмільйона пакетів, які може вільно використовувати будь-який Node.js-розробник.

3.5 Система управління базами даних MongoDB

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційних баз даних.

З давніх часів було звичайною справою зберігати всі дані в реляційних базах даних (MS SQL, MySQL, Oracle, PostgresSQL). При цьому було не так важливо, а чи підходять реляційні бази даних для зберігання даного типу даних чи ні.

На відміну від реляційних баз даних MongoDB пропонує документо-орієнтовану модель даних, завдяки чому MongoDB працює швидше, має кращу масштабованість, її легше використовувати.

Але, навіть враховуючи всі недоліки традиційних баз даних і гідності MongoDB, важливо розуміти, що завдання бувають різні і методи їх вирішення бувають різні. В якійсь ситуації MongoDB дійсно поліпшить продуктивність вашої програми, наприклад,

якщо треба зберігати складні за структурою дані. В іншій же ситуації краще буде використовувати традиційні реляційні бази даних. Крім того, можна використовувати змішані підхід: зберігати один тип даних в MongoDB, а інший тип даних - в традиційних БД.

Вся система MongoDB може представляти не тільки одну базу даних, що знаходиться на одному фізичному сервері. Функціональність MongoDB дозволяє розташувати кілька баз даних на декількох фізичних серверах, і ці бази даних зможуть легко обмінюватися даними і зберігати цілісність.

Одним з популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON.

BSON дозволяє працювати з даними швидше: швидше виконується пошук і обробку даних. Хоча треба зазначити, що BSON на відміну від зберігання даних в форматі JSON має невеликий недолік: в цілому дані в JSON-форматі займають менше місця, ніж в форматі BSON, з іншого боку, даний недолік окупається швидкістю.

MongoDB написана на C ++, тому її легко перенести на найрізноманітніші платформи. MongoDB може бути розгорнута на платформах Windows, Linux, MacOS, Solaris. Можна також завантажити вихідний код і самому скомпілювати MongoDB, але рекомендується використовувати бібліотеки з офіційних сайтів[13].

Якщо реляційні бази даних зберігають рядки, то MongoDB зберігає документи. На відміну від рядків документи можуть зберігати складну за структурою інформацію. Документ можна уявити як сховище ключів і значень.

Ключ являє просту мітку, з яким асоційоване значення.

Однак при всіх відмінностях є одна особливість, яка зближує MongoDB і реляційні бази даних. У реляційних СУБД зустрічається таке поняття як первинний ключ. Це поняття описує якийсь стовпець, який має унікальні значення. У MongoDB для кожного

документа є унікальний ідентифікатор, який називається _id. І якщо явно не вказати його значення, то MongoDB автоматично згенерує для нього значення.

Кожному ключу зіставляється певне значення. Але тут також треба враховувати одну особливість: якщо в реляційних базах є чітко окреслена структура, де є поля, і якщо якесь поле не має значення, йому (в залежності від налаштувань конкретної бд) можна привласнити значення NULL. У MongoDB все інакше. Якщо якомусь ключ не відповідає якесь значення, то цей ключ просто опускається в документі і не вживається.

Якщо в традиційному світі SQL є таблиці, то в світі MongoDB є колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об'єкти, то в колекції можуть містити найрізноманітніші об'єкти, що мають різну структуру і різний набір властивостей.

Система зберігання даних в MongoDB представляє набір реплік. У цьому наборі є основний вузол, а також може бути набір вторинних вузлів. Всі вторинні вузли зберігають цілісність і автоматично оновлюються разом з оновленням головного вузла. І якщо основний вузол з якихось причин виходить з ладу, то один з вторинних вузлів стає головним.

Відсутність жорсткої схеми бази даних і в зв'язку з цим потреби при щонайменшій зміні концепції зберігання даних оновлювати цю схему значно полегшують роботу з базами даних MongoDB і подальшим їх масштабуванням. Крім того, економиться час розробників. Їм більше не треба думати про обновлення бази даних і витрачати час на побудову складних запитів.

3.6 Платформа для розробки веб-додатків Angular

AngularJS - JavaScript-фреймворк з відкритим вихідним кодом. Призначений для розробки односторінкових додатків. Його мета - розширення браузерних додатків на основі MVC-шаблону, а також спрощення тестування і розробки.

Фреймворк працює з HTML, що містить додаткові атрибути, які описуються директивами, і пов'язує введення або виведення області сторінки з моделлю, яка представляє собою звичайні змінні JavaScript. Значення цих змінних задаються вручну або витягуються з статичних або динамічних JSON-даних [14].

AngularJS спроектований з переконанням, що декларативне програмування найкраще підходить для побудови призначених для користувача інтерфейсів і опису програмних компонентів, в той час як імперативне програмування відмінно підходить для опису бізнес-логіки. Фреймворк адаптує і розширює традиційний HTML, щоб забезпечити двосторонню прив'язку даних для динамічного контенту, що дозволяє автоматично синхронізувати модель і уявлення. В результаті AngularJS зменшує роль DOM-маніпуляцій і покращує тестувальність.

За допомогою директив AngularJS можна створювати призначені для користувача HTML-теги і атрибути, щоб додати поведінку деяких елементів.

Ng-app - оголошує елемент кореневих для додатка.

Ng-bind - автоматично замінює текст HTML-елемента на значення переданого вираження.

Ng-model - те ж, що і ng-bind, тільки забезпечує двостороннє зв'язування даних. Зміниться вміст елемента - ангуляр змінить і значення моделі. Зміниться значення моделі - ангуляр змінить текст всередині елемента.

Ng-class - визначає класи для динамічного завантаження.

Ng-controller - визначає JavaScript-контролер для обчислення HTML-виразів відповідно до MVC.

Ng-repeat - створює екземпляр DOM для кожного елемента з колекції. [14]

Ng-show і Ng-hide - показує або приховує елемент, в залежності від значення логічного виразу.

Ng-switch - створює екземпляр шаблону з безлічі варіантів, в залежності від значення виразу.

Ng-view - базова директива, відповідає за обробку маршрутів, які приймають JSON перед відображенням шаблонів, керованих зазначеними контролерами.

Ng-if - видаляє або створює частина DOM-дерева в залежності від значення виразу. Якщо значення виразу, призначеного ngIf, рівне false, об'єкт був видалений з DOM, інакше - знову клонований елемент вставляється в DOM.

Двостороння зв'язування даних в AngularJS є найбільш примітною особливістю, і зменшує кількість коду, звільняючи сервер від роботи з шаблонами. Замість цього, шаблони відображаються як звичайний HTML, наповнений даними, що містяться в області видимості, визначеної в моделі. Сервіс $ scope в Angular стежить за змінами в моделі і змінює розділ HTML-вирази в поданні через контролер. Крім того, будь-які зміни в поданні відображаються в моделі. Це дозволяє обійти необхідність маніпулювання DOM і полегшує ініціалізацію і прототипування веб-додатків.

3.7 Розмітка веб-сторінки HTML

Якщо в традиційному світі SQL є таблиці, то в світі MongoDB є колекції. І якщо в реляційних БД таблиці зберігають однотипні жорстко структуровані об'єкти, то в колекції можуть містити найрізноманітніші об'єкти, що мають різну структуру і різний набір властивостей.

Стандартна мова розмітки веб-сторінок в Інтернеті — HTML. Більшість веб-сторінок створюються за допомогою мови HTML [15]. Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді. HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту. Попри те, що HTML — штучна комп'ютерна мова, вона не є мовою програмування. HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок.

HTML впроваджує засоби для:

– створення структурованого документу шляхом позначення
структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;
– отримання інформації із Всесвітньої мережі через гіперпосилання;
– створення інтерактивних форм;
– включення зображень, звуку, відео, та інших об'єктів до тексту.

3.8 Каскадні таблиці стилів CSS

Каскадні таблиці стилів — спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі. CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі — сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо [16].

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації.

CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки. Одна з головних переваг — можливість розділити зміст сторінки (або контент, наповнення, зазвичай HTML, XML або подібна мова розмітки) від вигляду документу (що описується в CSS).

Один і той самий HTML або XML документ може бути відображений по-різному залежно від використаного CSS. Головні переваги CSS: інформація про стиль для усього

сайту або його частин може міститися в одному .css-файлі, що дозволяє швидко робити зміни в дизайні та презентації сторінок, різна інформація про стилі для різних типів користувачів: наприклад великий розмір шрифту для користувачів з послабленим зором, стилі для виводу сторінки на принтер, стиль для мобільних пристроїв, сторінки зменшуються в об'ємі та стають більш структурованими, оскільки інформація про стилі відділена від тексту та має певні правила застосування і сторінка побудована з урахуванням їх, прискорення завантаження сторінок і зменшення обсягів інформації, що передається, навантаження на сервер та канал передачі. Досягається за рахунок того, що сучасні браузери здатні кешувати (запам'ятовувати) інформацію про стилі і використовувати для всіх сторінок, а не завантажувати для кожної.

3.9 Скриптова метамова SASS

SASS це мова сценаріїв, котра інтерпретується в каскадні таблиці стилів. Sass складається з двох синтаксисів. Оригінальний синтаксис, називається «відступ синтаксис», використовує синтаксис, схожий на Haml. Він використовує відступи для поділу блоків коду і перекладу рядка символів, окремі правила. Новий синтаксис, «SCSS», використовує блок форматування, як у CSS. Він використовує фігурні дужки для позначення блоків коду і крапка з комою для розділення рядків в межах блоку [17].

Переваги SASS над CSS:

– @import – дозволяє зберігати весь CSS код у різних файлах та за необхідності злити їх в один файл, для створення остаточного .css файлу;

– @вкладеність – вкладеність селекторів;

– $variables – використання змінних для збереження інформації;

– @математика – математичні операції для вирахування властивостей як чисел так і кольорів;

– @домішки (міксини) – для уникнення постійних повторень одинакових правил CSS;

– @цикли – Sass дозволяє перебір змінних за допомогою @for, @each та @while , які можуть бути використані для застосування різних стилів до елементів з однаковими ідентифікаторами або класами.

Висновки до розділу 3

Для реалізації поставленої задачі було використано сучасні технології, котрі є одними з найпопулярніших у світі, що забезпечує зручну підтримку проекту та надає можливість легкого масштабування. Для розробки серверної частини проекту було використано платформу Node.js та нереляційну систему управління базами даних MongoDB, які у сукупності забезпечують швидку обробку даних, легкість їхнього зберігання та доступу до даних. Для реалізації користувацького інтерфейсу був використаний найсучасніший JavaScript фреймворк Angular версії 6.9, за допомогою якого можна з легкість спілкуватися серверною частиною та відображати дані для користувача. Для розмітки веб-сторінки використано HTML, а для її опису CSS.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Розробка програмного продукту поділяється на дві частини: написання серверної частини та веб клієнта. Серверна частина написана на платформі Node.js та розподілена на два мікро-сервіси: спостерігач та головний мікро-сервіс для взаємодії з клієнтом. Клієнтська частина реалізована на JavaScript фреймворку Angular версії 6.9. Взаємодія клієнта з сервером відбувається за допомогою REST API, що дозволяє легко масштабувати клієнтську частину.

4.1 Взаємодія серверу та клієнта

Взаємодія клієнта з сервером відбувається через підключення до мережі інтернет, це означає що для роботи з програмним продуктом обов’язкова умова є підключення до мережі інтернет. Для взаємодії між клієнтом та сервером було використано принцип взаємодії Representational State Transfer (REST).

У випадку коли клієнт виконує перехід на будь-яку веб-сторінку, браузер посилає GET запит до серверу, сервер приймає запит і після його обробки відправляє відповідь, браузер отримує відповідь від серверу та відображає отриману інформацію клієнту.

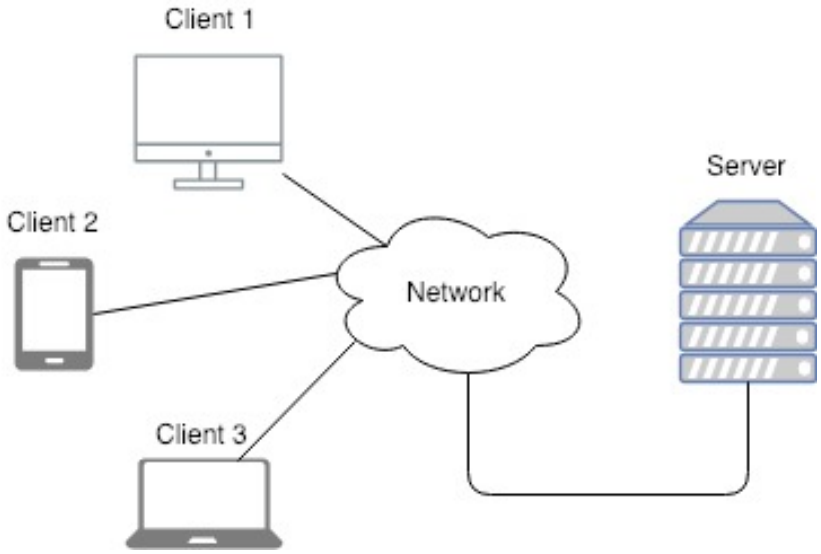

Рисунок 4.1 — Взаємодія між сервером та клієнтом

Для підвищення швидкості роботи серверної частини та надання можливості зручнішого масштабування проекту, серверну частину було розділено на два мікро-сервіси: спостерігач та головний сервіс взаємодії користувача з системою.

4.2 Сервіс “Спостерігач”

Для постійного спостереження за KNX-мережею було розроблено сервіс які працює не залежно від клієнтської частини та інших сервісі. Даний сервіс постійно підключений до KNX-мережі та записує у базу даних усі події, які відбуваються у мережі. Схема роботи сервісу зображена на рисунку 4.2.

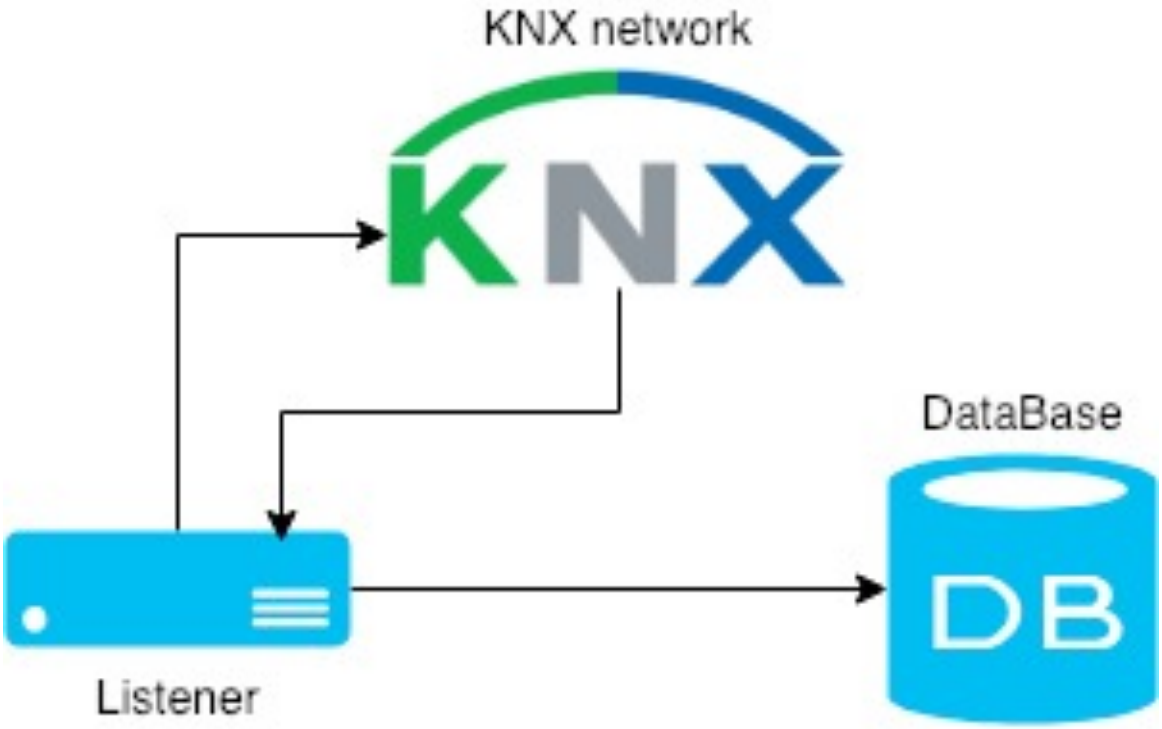

Рисунок 4.2 — Схема роботи сервісу “Спостерігач”

На рисунку 4.3 наведено приклад діаграми послідовності даного сервісу та роботи основних елементів підсистеми.

Першим кроком роботи сервісу є надсилання запиту до бази даних для отримання інформації про усі завантажені проекти. Після отримання результатів від бази даних, у випадку якщо в системі немає проектів, сервіс виводить повідомлення про це. У випадку коли сервіс отримує масив проектів, за допомогою циклічного оператора він встановлює підключення до KNX- мереж та чекає відповідь від мережі про стан підключення. Node.js надає можливість не блокувати виконання коду до відповіді від мережі а виконує дані підключення асинхронно. Після успішного підключення Спостерігач починає спостерігати за подіями котрі відбуваються у мережі та заносить дані про подію у базу даних з відповідним ідентифікатором проекту до якого відноситься дане підключення.

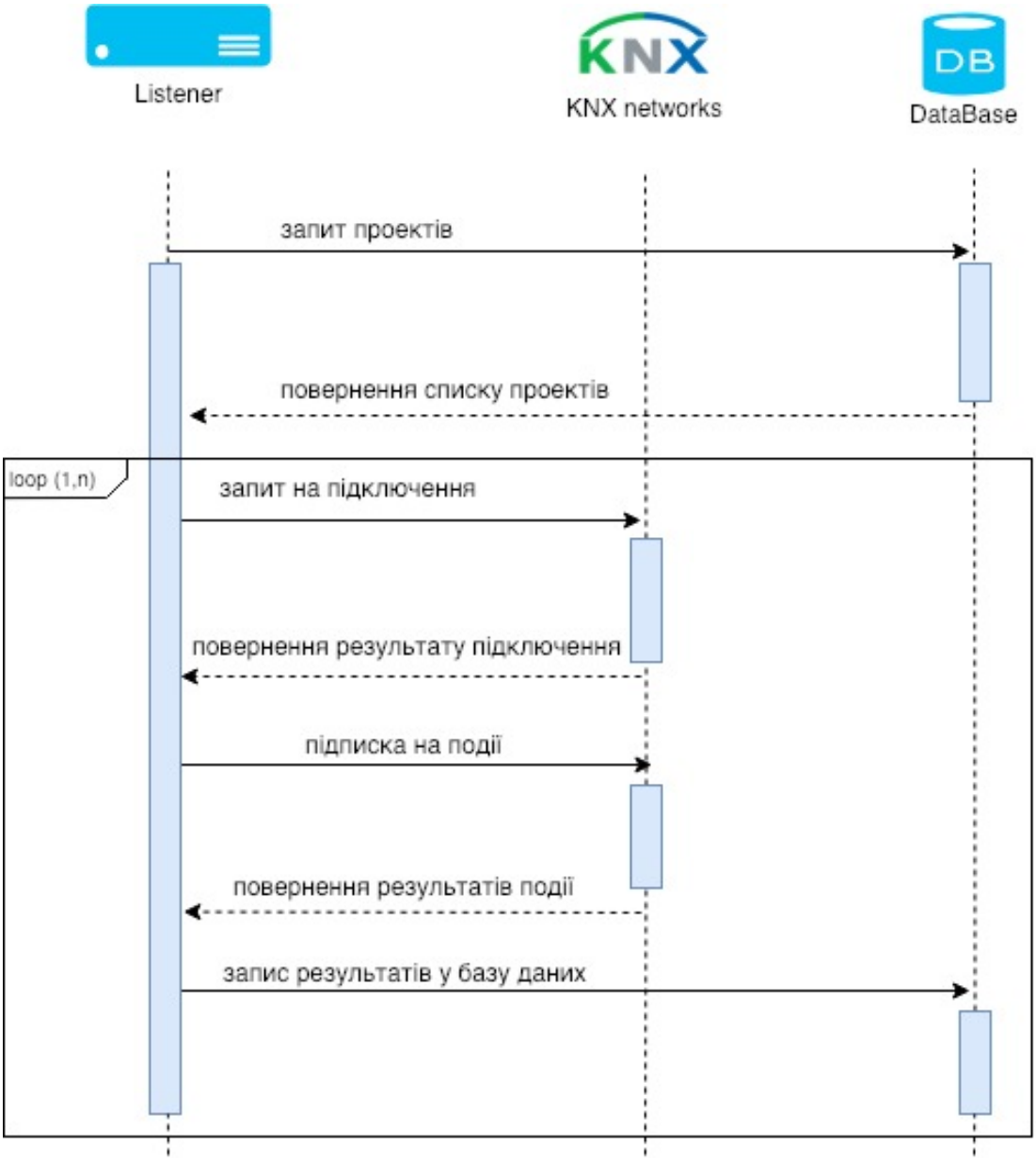

Рисунок 4.3 — Діаграма послідовності сервісу “Спостерігач”

За таким алгоритмом відбувається робота сервісу “Спостерігач”. Даний сервіс дозволяє не навантажувати інші елементи системи та працює абсолютно не залежно від них.

4.3 Сервіс взаємодії з клієнтом

Для взаємодії користувача з системою було розроблено окремий сервіс котрий забезпечує надсилання даних до користувача, отримання даних, обробка отриманих даних та внесення їх до бази даних. Оскільки користувач має можливість вибирати проекти для управління у даному сервісі реалізоване підключення до тієї KNX-мережі котру вибрав користувач та надає можливість управління елементами мережі та надсилання команд до неї. Схема роботи сервісу взаємодії з клієнтом зображено на рисунку 4.4.

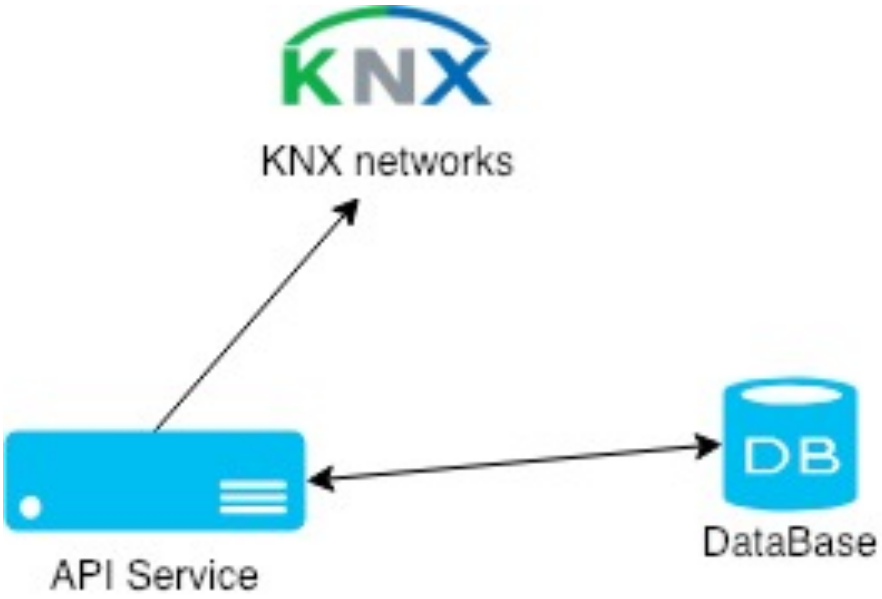

Рисунок 4.4 — Схема роботи сервісу взаємодії з клієнтом

API має можливість взаємодії з базою даних для зчитування та запису даних у неї та можливість надсилання команд до KNX- мережі у випадку коли користувачу необхідно виконати якусь дію з мережею.

Взаємодія клієнта з сервісом відбувається за допомогою HTTP протоколу передачі даних.

Висновки до розділу 4

Для взаємодії користувача з сервером було використано HTTP протокол передачі даних. Серверна частина розподілена на два мікро-сервіси: спостерігач за мережею та головний сервіс взаємодії з користувачем. Сервіс “Спостерігач” працює не залежно від інших елементів системи, відповідає за моніторинг усіх подій які відбуваються в системі та занесення даних до бази даних, на основі отриманих даних будуються діаграми роботи мережі. Сервіс взаємодії з клієнтом відповідає за надсилання даних до користувача, отримання даних, обробка, занесення до бази даних. Клієнтс.ька частина розроблена на JavaScript фреймворку Angular версії 6.9.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА

У даному розділі описано системні вимоги до ПК для забезпечення стабільної та коректної роботи розробленої системи, інструкцію користувача для її встановлення, а також детальну методику роботи користувача із програмною системою.

5.1 Системні вимоги

Для коректної роботи програмного продукту персональний комп'ютер маєвідповідати таким мінімальним системним та апаратним вимогам:

– встановлена операційна система Windows, Linux, MacOS;

– сучасний Браузер;

– процесор із тактовою частотою 2 ГГц або швидший – 32-розрядний (x86) або 64-розрядний (x64);

– оперативна пам'ять 2 гігабайт (ГБ) (для 32-розрядної версії) або 4 ГБ (для 64-розрядної версії).

5.2 Інсталяція системи

Вимоги до попередньо встановленого на машину програмного забезпечення:

– Node;

– Gulp;

– node.js пакет @angular/cli;

Для інсталяції серверної частини необхідно перейти до каталогу проекту та виконати команду "npm install" .

Для інсталяції мікро-сервісу що відповідає за спостереження мережі необхідно перейти у каталог мікро-сервісу та виконати команду “npm install” .

Для інсталяції клієнтської частини необхідно перейти у каталог проекту та виконати команду “npm install” .

Конкретні кроки, що повинні бути виконані для запуску системи в production-середовищі, залежать від особистої стратегії користувача (адміністратора).

Для початку роботи з системою необхідно:

1) Перейти у каталог мікро-сервісу та виконати команду “npm run start”
2) Перейти у каталог проекту що відповідає за серверну частину проекту та виконати команду “npm run start”
3) Перейти у каталог клієнтської частини та виконати команду “ng serve -o”

5.3 Сценарій роботи користувача з системою

При першому заході на веб сторінку системи у користувача немає завантажених проектів, тому він автоматично переходить на сторінку налаштувань (рисунок 5.1).

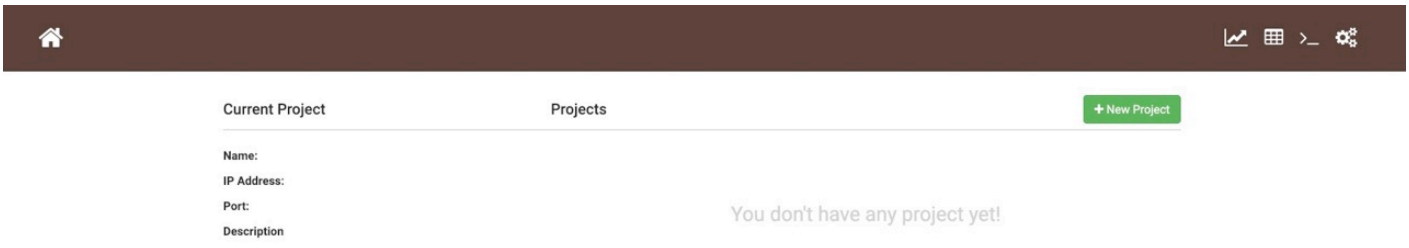

Рисунок 5.1 — Сторінка налаштувань

У випадку коли користувач буде намагатися перейти на будь яку іншу сторінку, але в систему не буде завантажено хоча б один проект, система автоматично буде блокувати перехід та повертати користувача на сторінку налаштувань. Для завантаження нового проекту необхідно натиснути на кнопку “New Project”, після чого користувач перейде на сторінку завантаження нового проекту (рисунок 5.2).

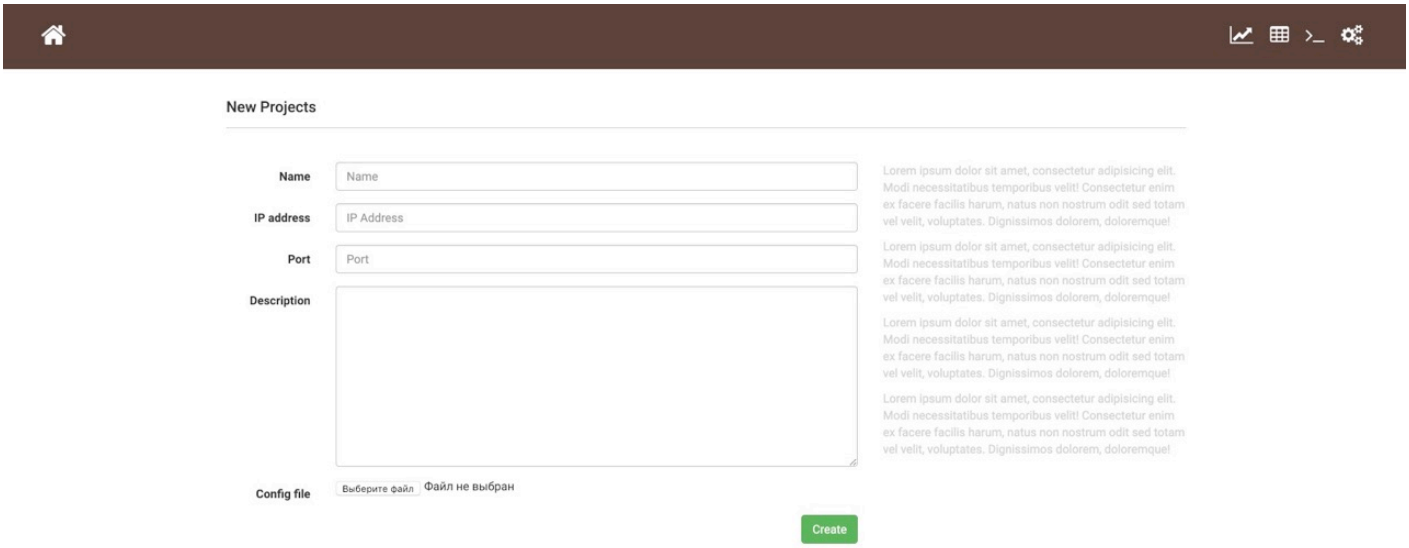

Рисунок 5.2 — Сторінка завантаження нового проекту

Для завантаження нового проекту користувачу необхідно заповнити усі поля та завантажити *.esf файл, який був експортований з EST. В результаті обробки даних у низу сторінки відображається таблиця зі структурою *.esf файлу (рисунок 5.3).

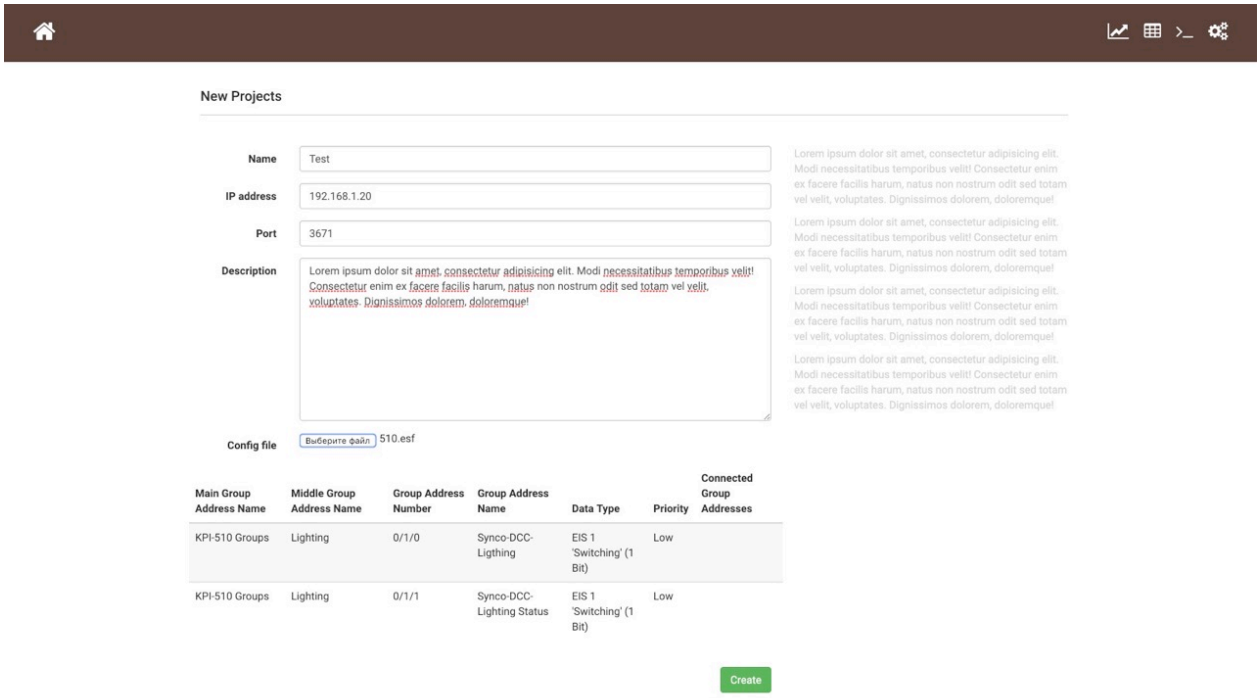

Рисунок 5.3 — Результат обробки даних

Для завантаження проекту в систему користувачу необхідно натиснути кнопку “Create”, після успішного завантаження проекту, система поверне користувача на сторінку налаштувань з існуючим списком проектів (рисунок 5.4). Користувач має можливість редагувати та видаляти проекти.

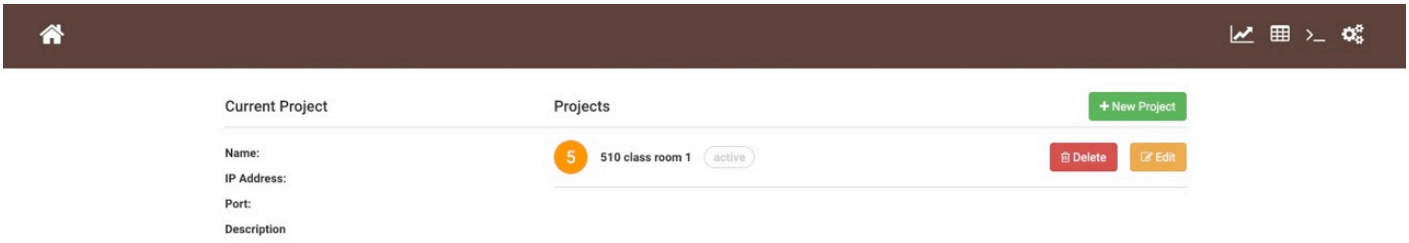

Рисунок 5.4 — Сторінка налаштувань зі списком проектів

Для управління проектом та підключення до KNX-мережі користувачу необхідно натиснути кнопку “active” яка знаходиться ліворуч від імені проекту у списку проектів. Після активації проекту клієнту надається можливість контролю та моніторингу хостів KNX-мережі. У проекту який є активним кнопка “active” зеленого кольору, з лівої від списку проектів відображається інформація про проект та назва проекту відображається у хедері сторінки що надає розуміння, на будь якій сторінці веб-сайту, який проект є активований. Сторінка з активованим проектом зображена на рисунку 5.5.

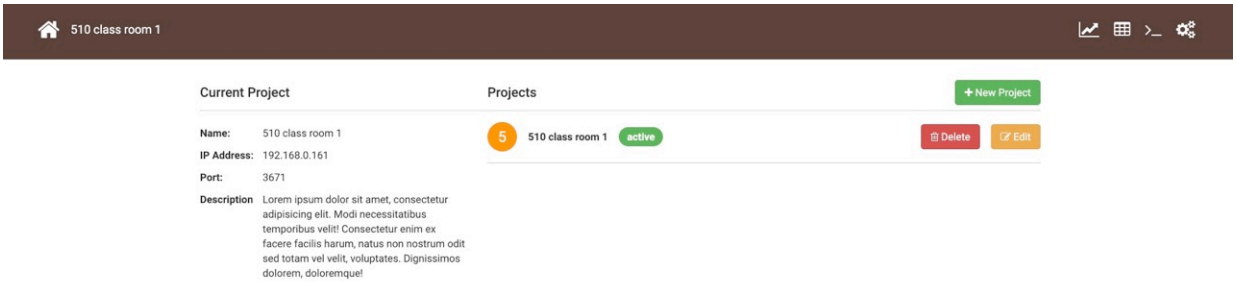

Рисунок 5.5 — Сторінка налаштувань з активованим проектом

Після активації проекту користувач має змогу перейти на головну сторінку системи (рисунок 5.6) з якої надається можливість контролю хостів KNX-мережі.

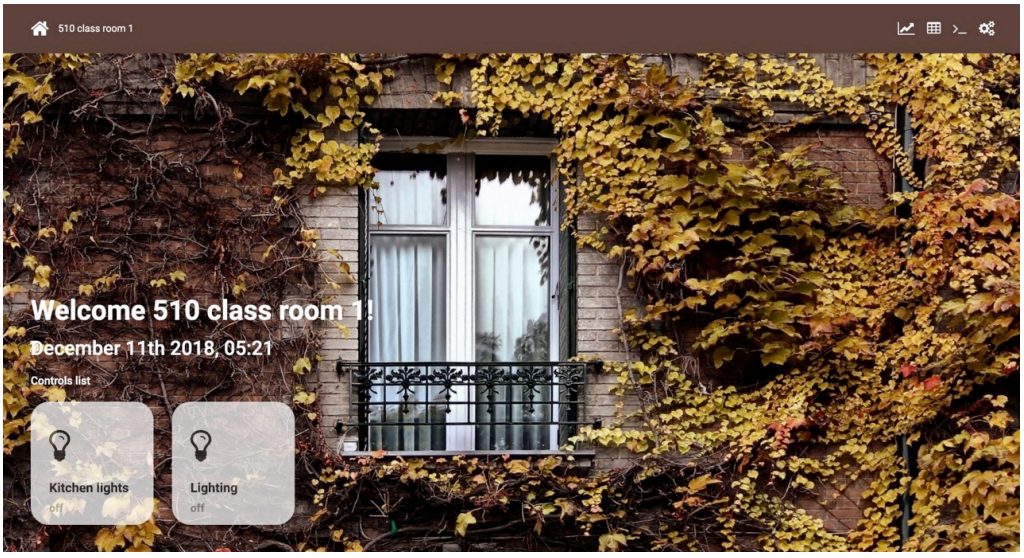

Рисунок 5.6 — Головна сторінка

У нижній частині головної сторінки розташований список елементів управління після натискання на елемент відбудеться зміна його стану (рисунок 5.7).

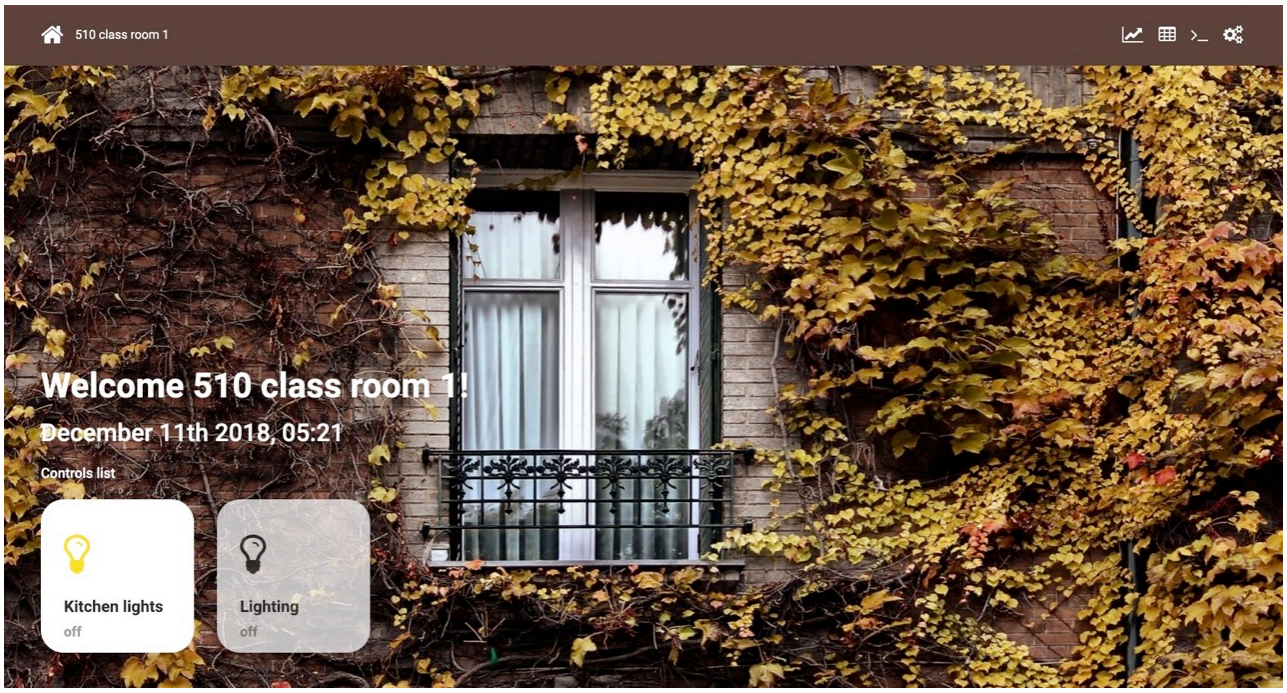

Рисунок 5.7 — Головна сторінка

Користувач має можливість моніторингу даних та перегляду їх у вигляді графіків та діаграм: загальний час роботи (рисунок 5.8) вісь Y – кількість годин, вісь X – стан, роботи та зміни станів елементів системи (рисунок 5.9).

Рисунок 5.9 — Графік роботи одного елементу

Користувач має можливість переглядати події у вигляді таблиць, для цього необхідно перейти на сторінку історії роботи системи (рисунок 5.10).

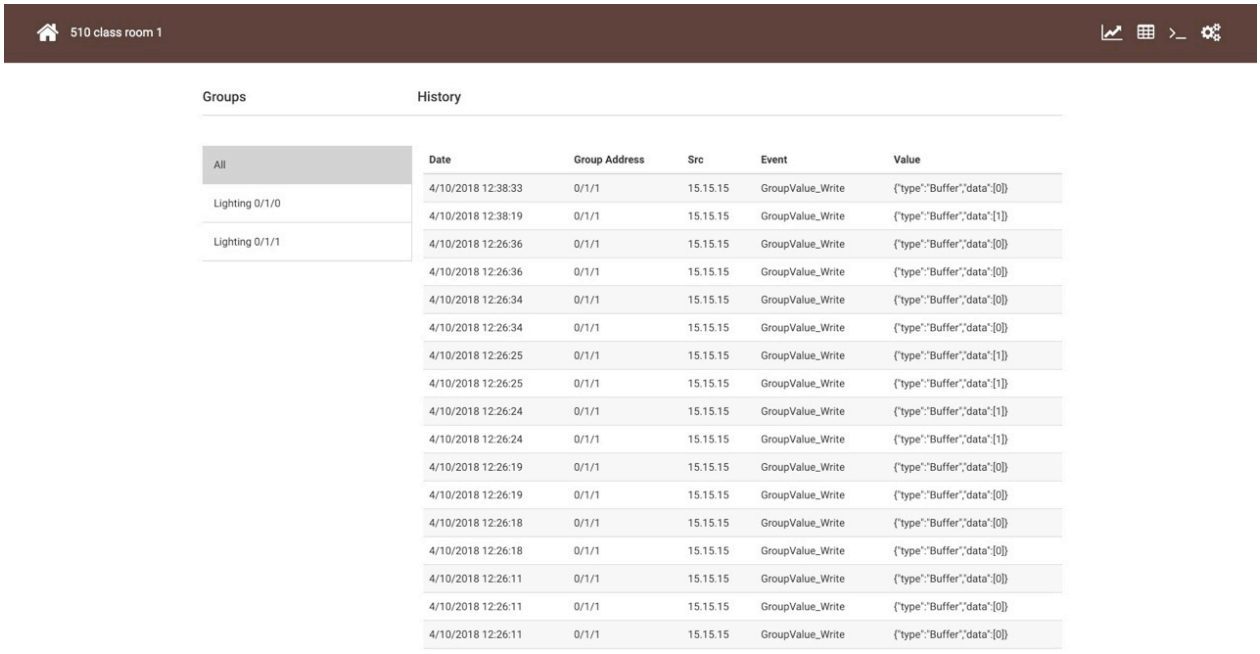

Рисунок 5.10 — Сторінка історії

Система надає можливість перегляду події які відбуваються в мережі у режимі реального часу, для цього необхідно перейти на сторінку термінал (рисунок 5.11).

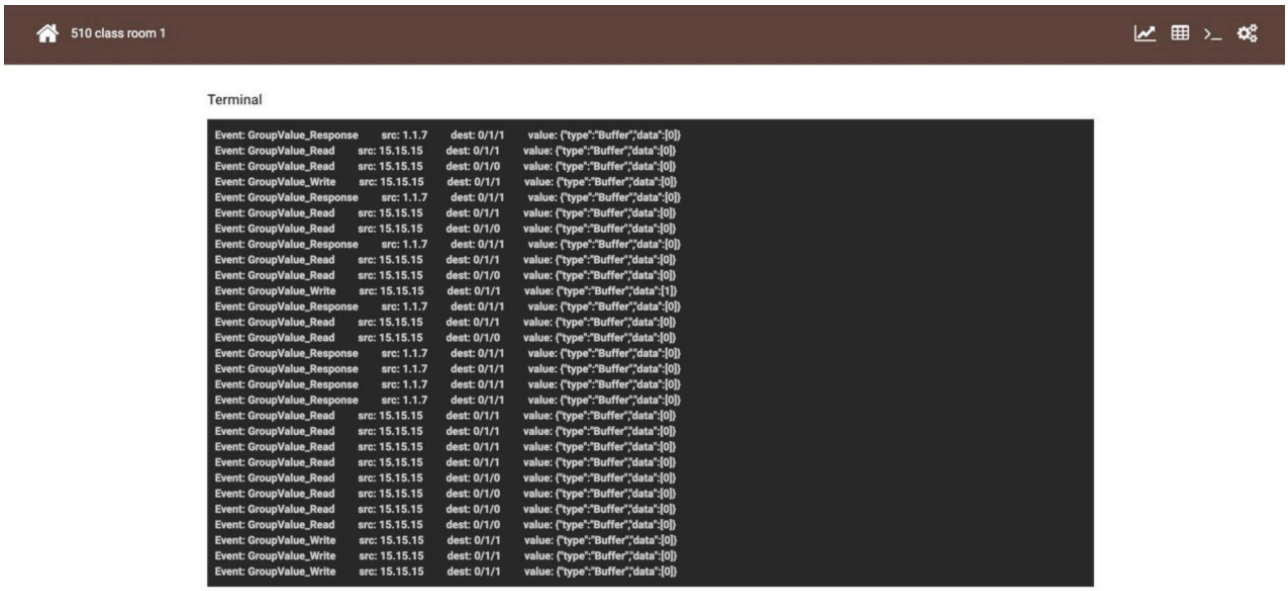

Рисунок 5.11 — Сторінка терміналу

Завдяки використанню сучасних технологій система мобільна версія сайту залишається повнофункціональної та однаково працює як на мобільних так і на комп’ютері. У результаті чого користувач має можливість доступу з різних пристроїв.

Висновки до розділу 5

Клієнтську частину реалізовано у вигляді веб-додатку, який є доступний з будь яких пристроїв, на які встановлено браузер. Застосунок має адаптивний дизайн, що забезпечує коректну роботу веб-додатку як з великих настільних комп’ютерів, так і з мобільних пристроїв.

6 РОЗРОБКА СТАРТАП ПРОЕКТУ

Ідея проекту полягає у розробці програмної реалізації людино-машиного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів. Дана система допоможе об’єднати програму для управління “розумним” приміщення та та програму для моніторингу усіх змін які відбуваються у системі. Система надає можливість управління управління різною кількістю не привязуючись до місця розташування користувача.

6.1 Опис ідеї стартап проекту

У підрозділі розглянуться наступні питання:

1. Ідея та її зміст.
2. Напрямки застосування.
3. Переваги для користувача.
4. Огляд існуючих аналогів.

Наступні пункти подаються у вигляді таблиці (таблиця 6.1) і відповідають на ряд поставлених питань.

Таблиця 6.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка програмної реалізації людино-машиного інтерфейсу	1. Візуалізація структури KNX-мережі	Можливість перегляду структури мережі та станів у яких знаходяться її хости.

для візуалізації структури KNX-мережі, контролю та моніторингу її хостів	2. Контроль та моніторинг хостів KNX-мережі	Можливість контролювати стани хостів та моніторинг подій що відбуваюлися у системі протягом певного періоду.

Для порівняння з пропозиціями конкурентних продуктів зроблено аналіз, який передбачає:

1. Створення списку властивостей та функцій ідеї.

2. Пошук і аналіз конкурентних продуктів чи проектів, товарів-замінників чи товарів-аналогів, які вже існують у відкритому доступі на ринку .

3. Проводиться порівняльний аналіз показників, для встановлення ідеї визначаються показники що мають: а) гірші значення(W, слабкі), б) аналогічні(N, нейтральні) значення, в) кращі значення(S, сильні)

Створено список потенційних переваг по відношенню до продуктів-конкурентів. Результат аналізу у таблиці 6.2.

Таблиця 6.2. Визначення характеристик ідеї проекту

Техніко-економічні характеристики ідеї	Продукція конкурентів		Слабкі (W), нейтральні (N) та сильні (S) сторони		
Назва продукту	Gideon	Apple HomeKit	W	N	S
Операційна система та версії	Android, IOS	MacOS, IOS			+

Системні вимоги	Мінімальні	Мінімальні			+
Мови програмування	Java, Objective c	Objective c		+	
Необхідність встановлення додаткового ПЗ	наявність АПК	наявність АПК		+	
Ціна	$600.00	$500.00			+

Аналог системи наразі функціонує та представлений як мобільний-додаток. Система підтримується на будь якій платформі. Вона не потребує особливих системних вимог.

На вітчизняному ринку аналогів створеній системі взагалі не виявлено.

6.2 Технологічний аудит ідеї проекту

Проведено аудит технології для проведення технічного аудиту ідеї проекту. за допомогою якої можна реалізувати ідею проекту. З аудите можна визначити чи доступні ці технології, яким чином можна вдосконалити технології [18]. Результат представлений у таблиці 6.3.

Таблиця 6.3. Технологічна здійсненність ідеї проекту

Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій

Розробка програмної реалізації людино-машиного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів	Середовище розробки JetBrains WebStorm	+	Доступна
	Node.js плагін KNX	+	Доступна

Для реалізації було обрано середовище розробки WebStorm від компанії JetBrains, а також Node.js плагін KNX для встановлення з взязку з KNX-мережею.

Обрані технології є доступними, не потребують допрацювань, а також безкоштовні та надають усі необхідні можливості для реалізації поставленої задачі.

6.3 Аналіз ринкових можливостей запуску стартап проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового

середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів. Результат досліджень наведено в таблиці 6.4.

Таблиця 6.4. Попередня характеристика потенційного ринку стартап-проекту

Показники стану ринку	Характеристика
Загальна потреба в продукції	Необхідна
Можливі річні обсяги випуску в натуральних показниках	Не обмежена
Річні обсяги випуску в вартісних показниках	1млн $
Динаміка ринку (якісна оцінка)	зростання
Наявність обмежень для входу	Для роботи необхідний доступ до мережі Інтернет
Специфічні вимоги до стандартизації та сертифікації	Відсутні
Середня норма рентабельності в галузі (або по ринку)	60 відсотків.

Висновком щодо аналізу потенційного ринку стартап проекту є оптимістичний проноз для виходу на ринок у найближчий час, оскільки наразі зацікавлених на ринку стає все більше і більше.

Як наслідком зростання попиту серед програмних продуктів, слід зробити більш зручну та автоматизовану систему для розробки програмного продукту та збільшити швидкість обчислення [19].

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 6.5).

Таблиця 6.5. Характеристика потенційних клієнтів стартап-проекту

Цільова аудиторія	Особливості поведінки споживачів	Вимоги споживачів до товару
Власники “розумних” приміщень	Візуалізація структури KNX-мережі та управління її хостами	доступна ціна; зручність і простота використання; мобільність
Проектувальники KNX-мереж	Можливість моніторингу хостів та необмежена кількість мереж для управління	- зручність і простота у використанні

Слідом за цим проводиться аналіз, який визначає клієнтів чи групу клієнтів, які є потенційними користувачами продукту. Слід провести аналіз моментів, які допоможуть чи перешкодять запровадити проект [20].

Результати представлені у таблицях 6.6 та 6.7 відповідно.

Таблиця 6.6. Фактори загроз

Фактор	Зміст загрози	Можлива реакція компанії
Поява конкурентів	Виникнення конкурентних проектів, які будуть	Оптимізація роботи, пришвидшення обчислень та додаткові можливості для користувача.

	фінансуватись компаніями-гігантами.	
Зміни тенденцій ринку	Виникнення спеціалізованих програм на подібну тему	Адаптація програмного продукту, який буду містити в собі максимальну кількість рішень, які є актуальними на ринку.
Економічний спад	У клієнтів буде поступово падати попит	Акції, скидки на певний термін.
Зниження репутації компанії	Ситуація при якій клієнти перейдуть до більш успішної компанії.	Рекламні компанії, покращення продукту, виправлення помилок, що призвели до псування іміджу, формування механізмів захисту від подальшого псування іміджу

Таблиця 6.7. Фактори можливостей

Фактор	Зміст загрози	Можлива реакція компанії
Невелика кількість конкурентів	Подібних програмних продуктів в нашій країні майже нема.	Піар свого проекту, реклама, презентації.
Зростання тенденцій ринку	Наразі зважаючи на ситуацію у країні попитом користуються технології, пов’язані з воєнною тематикою.	Швидка реклама та фокус на продажах, проведення зустрічей з клієнтами з презентаціями переваг проекту.

Підвищення іміджу компанії	Продукт є бажаним серед клієнтів	Активна реклама бренду

Наступним кроком слід визначити яка конкуренція буде очікувати та рівень конкуренції у товарів. Результати наведені у таблиці 6.8.

Таблиця 6.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії)
Тип конкуренції	Чиста Залежить від кількості конкурентів та якості	Підвищення якості продукту та зниження цін для встановлення балансу ціна-якість.
За рівнем конкурентної боротьби	Глобальна	Урахування культурних та регіональних чинників
За рівнем конкурентної боротьби	Локальна Конкуренція на вітчизняному ринку	Якщо конкурентних проектів на ринку немає, значить можна гратись з ціною на продукт.
Конкуренція за видами товарів	Створений товар може мати конкурентів, які пропонують аналогічний товар	Розширення функціональних можливостей, реклама для

		популяризації програмного продукту
За характером конкурентних переваг	Цінова	Можливе підвищення ціни на додатковий функціонал.
За інтенсивністю	Марочна	Реклама позитивних сторін в продукті, у випадку виходу на зовнішній ринок.

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 6.9) - за моделлю п’яти сил М. Портера, який вирізняє п’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції:

- конкурент, що вже є у галузі;
- потенційні конкуренти;
- наявність товарів-замінників;
- постачальники, що конкурують за ринкову владу;
- споживачі, які конкурують за ринкову владу.

Таблиця 6.9. Аналіз конкуренції в галузі за М.Портером

Складові галузі	Прямі конкуренти в галузі	Потенційні конкуренти	Клієнти	Товари-замінники
	Продукти аналоги	Кращі продукти, ширший функціонал	Налагодження комунікації з клієнтами	Немає

Висновки	Інтенсивність конкурентної боротьби з боку прямих конкурентів незначна	Наявні усі можливості входу на ринок. Конкурентів немає.	Необхідність клієнтської-бази, тому важливо знаходити можливості приваблення споживачів до власного продукту	Без обмежень

На основі аналізу конкуренції проведеного у таблиці 6.9, а також із враховуючи характеристики ідеї проекту (таблиця 6.2), вимог споживачів до товару (таблиця 6.5) та факторів маркетингового середовища (таблиці 6.6 і 6.7)на основі яких можна визначити та обґрунтувати перелік факторів конкурентоспроможності (таблиця 6.10).

Таблиця 6.10. Обґрунтування факторів конкурентоспроможності

Фактор конкурентоспроможності	Обґрунтування
Невелика кількість конкурентів на ринку	На початок розробки проекту не виявлено конкурентів на ринку в нашій країні.
Доступність створеного продукту (програмно)	Для застосування слід бути підключеним до мережі.
Легкість і простота використання	Зрозуміліший інтерфейс. Документація для використання.
Підключення до мережі Інтернет	Слід підключитись до мережі, що б викачати базу.
Потреба у постійному супроводі	Відсутня
Додаткові компоненти	Не потрібні

Після визначення факторів конкурентоспроможності можна визначити сильні та, навпаки, слабкі риси продукту. Вони наведені у таблиці 6.11.

Таблиця 6.11. Порівняльний аналіз сильних та слабких сторін проекту

Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
		–3	–2	–1	0	+1	+2	+3
Мала кількість / відсутність конкурентів	15				+			
Системні вимоги	20			+				
Простота використання	15				+			
Не потрібен супровід	5	+						

Для здійснення SWOT-аналізу необхідне відповідне інформаційне забезпечення, яке повинно включати: базу даних; методи та моделі, необхідні для SWOT-аналізу [22]. Методика SWOT-аналізу ґрунтується на підході, який дає змогу вивчати зовнішнє і внутрішнє середовище підприємства разом. За допомогою цієї методики можна встановити взаємозв’язки між силою та слабкістю.

Таблиця 6.12. SWOT-аналіз проекту

Сильні сторони (S):	Слабкі сторони (W):
розширений функціонал; ініціативна розробка; зручне маштабування;	мале фінансування; затрати на датчики;

гнучка політика керівництва;	
Можливості (О): вихід на міжнародні ринки; вдосконалення функціоналу; можливість іморту нових мереж;	Загрози (Т): незрозуміла тенденція попиту; поява конкурентів;

На основі SWOT-аналізу розробимо альтернативу ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок [23]. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів (таблиця 6.13).

Таблиця 6.13. Альтернативи ринкового впровадження стартап-проекту

Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Міжнародний ринок	Пошук клієнтів	Близько року
Розширення функціоналу	Пошук інвесторів	Від пів року.

Отже, програмний продукт чи проект слід випускати у світ лише після того, як буде проведений детальний аналіз ринку, конкурентів, можливих ризиків з якими можна зіштовхнутись.

6.4 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 6.14).

Таблиця 6.14. Вибір цільових груп потенційних споживачів

Опис цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в сегменті	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
Власники “розумних” приміщень	Наявні	Існує попит	Замала	Незначна
Проектувальники KNX-мереж	Наявні	Існує попит, проте менший у порівнянні з власника	Низька	Помірна

Провівши аналіз, та визначивши потенційних споживачів автори ідеї проекту починають вибирати цільові групи[24]. Саме для них і буде пропонуватись товар. Виходячи з аналізу буде визначена і вибрана потрібна стратегія [25].

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку, яка визначається у таблиці 6.15. За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Таблиця 6.15. Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку

Забезпечення ключових потреб клієнтів	Стратегія зосередження	Охоплення усіх ключових потреб клієнтів	Стратегія зосередженя
Орієнтація поточної моделі на ринок приватних підприємств	Стратегія концентрованого маркетингу	Надання товару кращих властивостей та розширення функціоналу	Стратегія спеціалізації (спирається на диференціацію)

Наступним кроком є вибір стратегії, яка подана у таблиці 6.16.

Таблиця 6.16. Визначення базової конкурентної поведінки

Чи є проект новий на ринку	Так
Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Так
Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Ні
Стратегія конкурентної поведінки	Стратегія заняття конкурентної ніші

Було орано стратегію заняття конкурентної ніші оскільки таку стратегію частіше за все використовують компанії, які є другими або третіми серед лідерів ринку.

На підставі вимог споживачів з обраних сегментів до постачальника та до програмного продукту необхідно розробити стратегію позиціонування (таблиця 6.17), що полягає у формуванні ринкової позиції, за яким споживачі мають ідентифікувати торгівельну марку або проект [23].

Таблиця 6.17. Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
Доступна ціна, простота і зручність використання	Стратегія диференціації	Доступність через ціну та технічні характеристики. Можливість гнучкого налаштування через веб-інтерфейс.	стандарти якості; метрики програмного забезпечення.

6.5 Розробка маркетингової програми стартап проекту

Роздивимось процес розробки маркетингової програми нашого страртап-проекту. Слід визначити і сформувати маркетингові концепції товару. У таблиці 6.18 наведено дослідження та їх результати щодо конкурентоспроможності продукту [22] .

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
Оцінка якості ПП	Спрощення бізнес процесів. Надання можливості планування доходів	Розрахункові показники, точність та достовірність яких можна оцінювати; самостійність програмної системи.

Далі необхідно розробити троьхрівневу маркетингову модель товару: уточняються ідея продукту, його фізичні складові та особливості процесу його надання (таблиця 6.19).

Таблиця 6.19. Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
Товар за задумом	Розробка програмної реалізації тонкого людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів.
Товар у реальному виконанні	Властивості/характеристики
	Реалізовано візуалізацію структури KNX-мережі у відповідності до завантаженого файлу конфігурації, здійснення контролю над її хостами та моніторинг подій у мережі.
Товар із підкріпленням	До продажу: стандартна розроблена
	Після продажу: додані додаткові можливості

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (таблиця 6.20).

Таблиця 6.20. Визначення меж встановлення ціни

Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни
50 – 1000 $	1000 – 5000 $	10 – 15 $

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 6.21)

Таблиця 6.21. Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Система повинна надаватися в режимах пробної версії та повної сплатити після закінчення випробувального строку	Проста та легкість в експлуатації	Веб-сайт	Вертикальна маркетингова система

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.22).

Таблиця 6.22. Концепція маркетингових комунікацій

Поведінка цільових клієнтів	Канали комунікацій цільових клієнтів	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення
Бажання отримати більше за менші гроші	Веб-сайт	Низька ціна Легкий і простий у використанні продукт	Довести, що програмний продукт оптимально працює з різними KNX-мережами та не потребує

			додаткових налаштувань

Висновки до розділу 6

У даному розділі було сформульовано ідею стартап проекту, проведено технологічний аудит, проаналізовано можливості ринку для запуску стартап проекту, розроблено ринкову стратегію та маркетингову програму. Переваги проекту в тому, що на ринку в нашій країні відсутні конкуренти, а значить конкуренції не виявлено. На разі попит на продукт великий, так як KNX-мережі набувають все більшої популярності для автоматизації будівель та приміщень.

Перспективи впровадження з огляду на потенційні групи користувачів, стан конкуренції та конкурентоспроможності проекту – прямі, і тільки доводять можливість впровадження, та не марну розробку створеного програмного продукту.

ВИСНОВКИ

Метою магістерської дисертації є розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів. Відповідно було проаналізовано основні принципи проектування інтерфейсу користувача та проблематику проектування інтерфейсів користувача. Ознайомлено зі стандартами KNX-мереж та програмними застосунками для їх проектування.

Враховуючи вимоги до зручності, безпеки та гнучкості сучасних будівель та враховуючи постійний технологічний розвиток та підвищення автоматизації систем забезпечення комфорту було розроблено людино-машинний інтерфейс для візуалізації структури мережі, контролю та моніторингу її компонентів.

Розроблена система забезпечує наступну функціональність:

— створення проектів для використання;
— редагування проектів у випадку коли відбулося масштабування мережі;
— надати можливість вибору мережі для управління;
— підключення до мережі;
— відображення структури KNX- мережі;
— посилання події до мережі;
— постійне спостереження за подіями які відбуваються в мережі;
— відображення історії користування;
— побудова графіків та діаграм користування елементами мережі;
— відображення подій мережі у реальному часі;

Серверну части системи було розділено на два мікросервіси: сервіс «Спостерігач» відповідає за моніторинг системи, та головний сервіс для взаємодії з клієнтом.

Клієнт реалізовано у вигляді веб-застосунку, що може бути доступний з будь-якого браузеру. Застосунок має адаптивний дизайн, що дозволяє користуватися ним як з великих настільних комп'ютерів, так і з мобільних телефонів.

Для реалізації поставленої задачі було використано сучасні технології, котрі є одними з найпопулярніших у світі, що забезпечує зручну підтримку проекту та надає можливість легкого масштабування. Для розробки серверної частини проекту було використано платформу Node.js та нереляційну систему управління базами даних MongoDB, які у сукупності забезпечують швидку обробку даних, легкість їхнього зберігання та доступу до даних. Для реалізації користувацького інтерфейсу був використаний найсучасніший JavaScript фреймворк Angular версії 6.9, за допомогою якого можна з легкість спілкуватися серверною частиною та відображати дані для користувача. Для розмітки веб-сторінки використано HTML, а для її опису CSS.

Було описано роботу з програмним продуктом. У розділі є інструкція роботи з програмою. Зазначено всі можливі функції, які може використовувати користувач програмного продукту.

Даний програмний продукт має суттєву перевагу над існуючими аналогами та конкурентами, які представлені на внутрішньому ринку. Продукт має шанси розвиватись, наповнюватись новим функціоналом, а отже було визначено основні маркетингові стратегії. Потенційними користувачами програмного продукту є морські інженерів-проектувальників KNX-мереж та користувачів які не є спеціалістами у даній області.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Технология KNX для систем автоматизации [електронный ресурс] – режим доступу: https://www.ixbt.com/home/knx-intro.shtml
2. Основні поняття ІК та засоби їх проектування [Електронний ресурс] — Режим доступу: http://elar.khnu.km.ua/jspui/bitstream/123456789/1415/2/Rozdil1.pdf
3. Тараненко К.Г. Автоматизо-ваний аналіз та оцінка зручності використ- тання програмних систем. Системний аналіз та інформаційні Системний аналіз та інформаційні технології / Тараненко К.Г., Гученко І.В. : матеріали 12-ї Міжнародної Науково-технічної конференції SAIT 2010, Київ, 25–29 травня 2010 р. / ННК “ІПСА” НТУУ “КПІ”. – К.: ННК“ІПСА” НТУУ “КПІ”, 2010. – 544 с.
4. John M. Carroll Human Computer Interaction (HCI) [Електронний ресурс] — Режим доступу: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed/human-computer-interaction-brief-intro
5. Стандарт KNX-мереж [Електронний ресурс] — Режим доступу: http://www.gudramaja.com/content/ru/296/%D0%92%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B2-%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82-KNX.html.
6. EST - Опис [Електронний ресурс] — Режим доступу: http://www.konnex-russia.ru/knx-standard/knx-tools/ets/.
7. OPC Export – Опис [Електронний ресурс] — Режим доступу: https://support.knx.org/hc/en-us/articles/115001823310-OPC-Export.
8. JetBrains WebStorm – Опис [Електронний ресурс] — Режим доступу: https://itpro.ua/product/jetbrains-webstorm/?tab=description.
9. David F. 12. JavaScript: The Definitive Guide / Flanagan David., 2011. –

1096 с. – (IEEE O'Reilly Media). https://www.typescriptlang.org/docs/home.html

10. TypeScript Docimentation– Опис [Електронний ресурс] — Режим доступу: https://www.typescriptlang.org/docs/home.html.
11 TypeScript Angular – Опис [Електронний ресурс] — Режим доступу:https://www.typescriptlang.org/docs/handbook/angular.html.
12 Tutorialspoint Node.js – Опис [Електронний ресурс] — Режим доступу: https://www.tutorialspoint.com/nodejs/nodejs_tutorial.pdf.
13. Руководство по MongoDB– Опис [Електронний ресурс] — Режим доступу: https://metanit.com/nosql/mongodb/.
14. Angular 6 [Електронний ресурс] – Опис [Електронний ресурс] — Режим доступу: https://angular.io.
15. The global structure of an HTML document – Опис [Електронний ресурс] — Режим доступу: https://www.w3.org/TR/html401/struct/global.html.
16. Современний учебник по CSS document – Опис [Електронний ресурс] — Режим доступу: https://idg.net.ua/blog/uchebnik-css.
17. Основи document – Опис [Електронний ресурс] — Режим доступу: https://sass-scss.ru/guide/
18. Черкасов Д. О., Сайбель Н. Ю. Стартап: характеристика понятия и этапы развития // Современное состояние и перспективы развития научной мысли: сборник статей Международной научно-практической конференции (25 мая 2015 г., г. Уфа) в 2 ч. – Уфа: АЭТЕРНА, 2015. – Ч. 1. – С.141-143.
19. Розроблення стартап-проекту: Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О. А. Гавриша. – Київ: НТУУ «КПІ», 2016. – 28 с.
20. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.

21. Тиль, П. От нуля к единице : как создать стартап, который изменит будущее / П. Тиль, Б. Мастерс; перевод с англ. – Москва : Альпина паблишер, 2015. – 188 с. Петруненков А. Оценка коммерческой привлекательности проекта [Електронний ресурс].– Режим доступу: http://www.techbusiness.ru/tb/archiv/number2/page01.htm
22. Каталог автоматизированных систем [Електроний ресурс]. – Электрон. дан. - Режим доступа: http://www.erponline.ru/software/open/
23. Филип Котлер, Роланд Бергер, Нильс Бикхофф. Стратегический менеджмент по Котлеру. Лучшие приемы и методы = The Quintessence of Strategic Management: What You Really Need to Know to Survive in Business. — М.: Альпина Паблишер, 2012. — 144 с.
24. Бариленко В. И. Бизнес-анализ как важный вид консалтинговых услуг // РИСК: Ресурсы, Информация, Снабжение, Конкуренция. — № 4. — 2012. — С.202-207.
25. Квашнин А. Как продвигать проекты коммерциализации технологий: серия методических материалов «Практические руководства для центров коммерциализации технологий» / М. Катешова, А. Квашнин, под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 52 с.

ДОДАТОК А

Апробації

Розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3268_18М

Аркушів 3

2018

ДОДАТОК Б

Акт впровадження

Розробка програмної реалізації людино-машинного інтерфейсу для візуалізації структури KNX-мережі, контролю та моніторингу її хостів

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ ТВ3268_18М

Аркушів 2

2018